7-1-1996

Image Compression and Representation using Wavelets

Shankar Moni
Purdue University School of Electrical and Computer Engineering

R. L. Kashyap
Purdue University School of Electrical and Computer Engineering

IMAGE COMPRESSION AND REPRESENTATION USING WAVELETS

SHANKAR MONI
R. L. KASHYAP

TR-ECE 96-12
JULY 1996

SCHOOL OF ELECTRICAL
AND COMPUTER ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, INDIANA 47907-1285

Image Compression and Representation using Wavelets

Shankar Moni and R. L. Kashyap

School of Electrical and Computer Engineering
Purdue University
1285 Electrical Engineering Building
West Lafayette, IN 47907-1285.

This work was supported in part by the Office of Naval Research under contract ONR N00014-91-J-4126 and in part by the NSF under Grant CDR 8803017 to the Engineering Research Center for Intelligent Manufacturing Systems.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

ABSTRACT

The focus of this research is to explore the structure present in the wavelet decomposition of natural images and use this structure for image compression and representation.

We first show empirically that the wavelet coefficients of a natural image, when sorted by magnitude, lie almost exactly on a curve of the form $\frac{1}{x^\alpha}$. This, in turn, helps us to demonstrate that natural images may exhibit self-similarity in the wavelet domain, although no such property may be apparent in the pixel domain.

In order to predict which scale-space positions may have large wavelet coefficients, we define a data structure called a "web". We use this together vvith a top-down adaptive search (TAS) algorithm that we propose. Using this, one can predict not only which scale-space positions have large wavelet coefficients, but also the magnitudes of the wavelet coefficients. Further, we are able to find a bound on the error in our prediction. We also show that this prediction scheme has the ability of automatically zooming in onto the edges in the image. Additionally, this method of predicting scale-space positions yields a number (we call it y) which is indicative of the amount of structure present in an image. Finally, we derive a theoretical bound on the value y can attain when the TAS algorithm is applied to an autoregressive process. This theoretical bound further supports the empirical evidence that y is indicative of the amount of structure in the image.

We use the prediction scheme to construct an image compression algorithm. The image compression algorithm is comparable in performance to the best available algorithms and also has low computational requirements. Further, the algorithm allows the user to control both the mean square error as well as the number of non-zero coefficients in the representation.

Finally, we use these properties to formulate a stochastic process to model an image and explore various theoretical properties of the stochastic process.

1. INTRODUCTION

1.1 Introduction

Image compression is an important field of research that has been studied for nearly three decades now. Compression of images has numerous applications in diverse areas such as high definition television, videophones, medical imaging, on-line product catalogs and other multimedia applications. Another important application is browsing, where the focus is on getting high compression. For many years, the most popular image compression technique was based on the discrete cosine transform (DCT) [1]. In the recent past, wavelets have emerged as an important technique for image compression (see chapter 2 for a review of wavelets).

The key idea employed in image compression is that there is statistical structure present in an image. Virtually all image compression algorithms exploit this statistical structure to devise a compressed representation of an image. In recent years, many researchers have proposed compression algorithms based on the wavelet decomposition of an image [2, 3, 4, 5, 6, 7]. Since these wavelet-based algorithms have proved to be very successful, it is of interest to undertake a detailed study of the statistical structure present in the wavelet decomposition of an image.

The most obvious statistical structure present in natural[1] images is that neighboring pixels are highly correlated except across edges. Another simple example of statistical structure is that it is possible to fit a probability density to the wavelet coefficients [8, 9]. A different form of statistical structure observed by Shapiro is the ability to predict which wavelet coefficients may be small in magnitude [3]. Other

[1] In this thesis, "natural images" (as opposed to synthetic images) refers to images that have been captured with a camera.

forms of statistical structure used are self-similarity in images [10, 11, 12] and modeling of the coefficients of the discrete cosine transform (DCT) of an. image [13].

Given a goal (such as image compression), it is of interest to highlight the statistical structures present that help in attaining this goal. This is usually done by constructing a model of the image that includes the necessary statistical structure. Early image models were based on a Markov mesh and were used for goals such as pattern classification [14], image estimation [15, 16] and texture segmentation [17]. These models have been generalized to multiscale Markov mesh models [18] and have been successfully applied towards various goals such as image segmentation [19, 20], image restoration [21, 22] and automated inspection [23]. Recently, Ran and Farvardin have proposed an image model that is based on the psychovisual properties of the human visual system [24] which they use in image compression [25]. With the recent success of wavelets in image compression, it is clear that wavelet based image models are an indispensible tool for understanding images and formulating better image compression algorithms.

1.2 Image Compression using Wavelets

We now review a few basic ideas about representing an image using wavelets. A more detailed introduction to wavelets may be found in chapter 2. Let $\psi(x, \mathrm{y})$ be a two-dimensional wavelet and let $\psi_d(x, \mathrm{y})$ denote a dilated and translated version of ψ. The subscript d is called a scale-space position. An image f can now be represented as

$$f(x,y) = \sum_{d \in D} c_d \, \psi_d(x,y) \tag{1.1}$$

In wavelet-based image compression, it is necessary to choose a subset D of scale-space positions which will be used to represent the image. Then one has to encode the scale-space position, the magnitude and the sign of each chosen coefficient. The scale-space position simply indicates which dilate and translate of the wavelet (i.e. which basis function) was used.

Some algorithms use vector quantization to perform this encoding [2, 26]. Vector quantization exploits the fact that the data to be encoded exhibits correlation. Thus, in a loose sense, one may say that these methods predict the magnitudes and signs of the coefficients. However, these methods do not directly predict the magnitudes and signs from the image data while coefficients are being chosen. A different approach is the zero-tree algorithm of Shapiro **[3]** which only attempts to predlict the positions of the coefficients. In fact, Shapiro suggests that a large percentage of the bit budget is used for encoding the positions. Katsavounidis et al. [5] extend Shapiro's method to include vector quantization. They also remark that the computational complexity increases significantly when vector quantization is used. Therefore, there is still a need for a method which can predict the positions and magnitudes and do so in a computationally efficient manner.

There are different methodologies to decide which wavelet coefficients are to be retained in the representation. Many algorithms choose the coefficients with the largest magnitudes as the ones to be retained [3, 5]. Other algorithms try to utilize the psychophysics of the human visual system when choosing which coefficients to retain [2, 6]. The coefficients are first weighted according to the sensitivity of the human eye to the energy of the coefficients. Then the ones with largest magnitude are chosen. The computational requirement of all these methods includes (i) computation of the complete wavelet transform, and (ii) ordering (or thresholding) of the wavelet coefficients.

1.3 Measures of Performance

There are several criteria to measure the performance of a compression algorithm. The most common way to attempt to measure visual quality is using the peak signal to noise ratio (PSNR) [27] which is based on the mean-squared error. Although many researchers admit that the mean square error is not the best measure of visual quality [27, 28], it is commonly used because of its simplicity. However, DeVore et al. [4] have suggested a different measure, which is the number of non-zero coefficients retained

in the compressed representation. This measure emphasizes the idea, that it is easier for the human eye to discern the presence of non-zero coefficients than to precisely estimate the value of a coefficient.

Apart from visual quality, another measure of performance is the computational requirement of the algorithm. Computationally efficient image compression algorithms are very important in in applications requiring real-time compression.

1.4 Contributions of this thesis

To compress an image, we would like to choose the subset D of !scale-space positions (in equation (1.1)) that have large coefficients. In the absence of any statistical information, the description of the set D may consume a large fraction of the bit-budget available for encoding. It is therefore of interest to be able to predict the set D. To do this, we have devised a top-down adaptive search (TAS) algorithm which adaptively chooses scale-space positions in a greedy manner.

In order to evaluate the performance of the TAS strategy, we need to see how the coefficients look when they are actually chosen in decreasing order of magnitude. We find a somewhat surprising result: The wavelet coefficients of an image, when sorted in decreasing order of magnitude, lie almost precisely on a curve of the form $y = \frac{C}{(x+\triangle)^\alpha}$. It turns out that the wavelet coefficients, when arranged in the order prescribed by the TAS algorithm, also lie approximately on the same curve. Further, we are able to find an bound on the error in using the TAS algorithm (as opposed to a true sort by magnitude).

We now briefly outline the contributions of this thesis. We

- show empirically that the wavelet coefficients of an image, when sorted in decreasing order of magnitude, lie almost precisely on a curve of the form $y = \frac{C}{(x+\triangle)^\alpha}$. Further, we show empirically that certain natural images exhibit a form of self-similarity in the wavelet domain although no such property may be apparent in the pixel domain.

- define a top-down adaptive search (TAS) algorithm which provides an adaptive, structured methodology for selecting dilates and translates of the wavelet. The TAS algorithm is useful because it enables us to predict (i) the: *positions* of the significant coefficients and (ii) the *magnitude* of the coefficient. We use the TAS algorithm together with a data structure called a "web" which. we define.

- show that the TAS algorithm automatically zooms in onto the edges present in the image.

- show that the TAS algorithm yields a number (we call it y), which is indicative of the amount of structure present in the image. We also derive a theoretical upper bound on the value of y that will result when the algorithm is applied to an autoregressive process. This theoretical bound further supports the empirical evidence that y is indicative of the amount of structure in the image.

- propose an algorithm for image compression that

  - allows the user to control both the mean square error and number of coefficients,
  - has performance comparable to the best available algoritlhms,
  - has low computational requirements,
  - yields a conditionally embedded [3] bitstream,

- formulate a wavelet-based stochastic process which can serve as an image model. For the stochastic process, we

  - prove its existence.
  - derive necessary and sufficient conditions for the process to have finite energy.
  - prove that it provides infinite details everywhere.

This stochastic process is based on the empirical observations. It can serve as a theoretical framework to perform formal studies on image compression algorithms and also opens up the possibility of new theoretical and practical developments in the field.

The results in this thesis is based on the empirical observations from a study on the database of 20 images shown in figure (1.1). Each image was analysed using four different wavelets: Haar [29, pg 277], Daubechies D-4 [30, pg 129], spline-variant [29, pg 279] and 2nd order spline wavelet [29, pg 277]. We realise that it is possible to find both synthetic and natural images for which the corresponding empirical observations will not hold. However, as can be seen from figure (1.1), the observations appear to hold for a wide class of images of interest.

1.5 Organization of the thesis

The remainder of this thesis is organized as follows. In chapter 2, we provide a brief review of wavelets. In chapter 3, we examine various properties of the wavelet coefficients of natural images. We also introduce the TAS algorithm for choosing scale-space positions. Further, we introduce the data structure called the web, which is used by the TAS algorithm. We discuss various empirical properties of wavelet coefficients that result from using the TAS algorithm with the web. Chapter 4 deals with image compression using these various empirical properties. Chapter 5 discusses the use of these properties to construct a stochastic process for image representation. Conclusions are presented in chapter 6, which is followed by the appendix containing proofs of theorems.

Fig. 1.1. Database of images used for the studies in this thesis. Names of the images in raster-scan order: Lenna, Baboon, Car, Couple, F16, Fields, Hotel, Airport, Mustang, Peppers, Sailboat, Stream, Widget, Bank, Uchart, Picnic, Einstein, Oval, Oval+noise, Noise.

2. REVIEW OF MULTIRESOLUTION ANALYSIS AND WAVELETS

In this chapter, we review the basic principles of multiresolution analysis and wavelets. We only review those ideas that are essential for the following chapters since there are many sources to read about other developments in the field. There are several recent books on the subject such as the ones by Vetterli and Kovacevic [31], Strang and Nguyen [32], Walter [33], Chui [30], Daubechies [29] and Meyer [34]. Several expository papers have also been published on the subject [35] [36] [37].

We let R denote the set of real numbers and Z denote the set of integers. The set of all square integrable functions is denoted by $L^2(R)$ and the set of square summable sequences is denoted by $\ell^2(\mathbf{Z})$.

2.1 Multiresolution Analysis

A multiresolution analysis of $L^2(R)$ is defined as a sequence of closed subspaces V_j of $\mathcal{L}^2(\mathbf{R})$, $j \in Z$, with the following properties [38]

1. $V_j \subset V_{j+1}$
2. $v(x) \in V_j \Leftrightarrow v(2x) \in V_{j+1}$
3. $v(x) \in V_0 \Leftrightarrow v(x+1) \in V_0$
4. $\bigcup_{j=-\infty}^{\infty} V_j$ is dense in $L^2(R)$ and $\bigcap_{j=-\infty}^{\infty} V_j = \{\mathbf{0}\}$
5. A scaling function $\phi \in V_0$ with a nonvanishing integral exists such that the collection $\{\phi(x - I) : I \in Z)$ is a Riesz basis [30, page 71] for V_0.

From *(2)* and (5), it is obvious that $\{\phi(2x - \mathrm{k}), \boldsymbol{k} \in \mathbf{Z}\}$ spans V_1. Since $V_0 \subset V_1$, it follows that there exists a sequence $\{p_k\} \in \ell^2(\mathbf{Z})$ such that

$$\phi(x) = \sum_{k=-\infty}^{\infty} p_k \phi(2x - k) \tag{2.1}$$

The above equation is called the dilation equation or the two-scale relation. In the Fourier domain, this can be written as

$$\Phi(\omega) \;=\; \tfrac{1}{2} P(e^{-j\omega/2}) \Phi(\frac{\omega}{2}) \tag{2.2}$$

where Φ denotes the Fourier transform of ϕ and $P(z)$ is the z-transform of the sequence $\{p_k\}$.

An Example: It is well known that splines satisfy the two-scale relation. The book by Chui [30] deals extensively with spline-based multiresolution analyses. We let $\chi_{(0,1]}$ denote the indicator function of $(0, 1]$. The zeroth order spline is defined as

$$\phi^0(x) = \chi_{(0,1]}(x) \tag{2.3}$$

and the mth order spline is defined recursively by the relation

$$\phi^m(x) = [\phi^{m-1} * \phi^0](x) \tag{2.4}$$

where $*$ denotes the convolution operator. Splines obey a finite two scale relation which is given by

$$\phi^m(x) = \sum_{k=0}^{m+1} 2^{-m} \binom{m+1}{k} \phi^m(2x - k)$$

This relation can be used to generate a multiresolution analysis because splines satisfy the other requirements of a scaling function outlined in properties *1* through 5 above.

2.2 Wavelets

Since $V_j \subset V_{j+1}$, the space V_{j+1} can be expressed as

$$V_{j+1} \;=\; V_j \bigoplus W_j \tag{2.5}$$

where $\oplus$ denotes an orthogonal sum. That is, W_j is the largest subspace of V_{j+1} that is orthogonal to V_j. The spaces W_j are called the "detail" spaces because they contain the details left out by projection of a function onto V_j. It has been shown [30] [29] that for any multiresolution analysis, there exists a function ψ, called the wavelet, whose translates span the detail space. That is,

$$W_j = \text{span}\{\psi(2^j x - \text{k}) \ : \text{k} \in \text{Z}).$$

Since $W_j \subset V_{j+1}$, the wavelet ψ can be expressed as a weighted sum of shifts of $\phi(2x)$, i.e.

$$\psi(x) = \sum_k q_k \phi(2x - k) \tag{2.6}$$

for some square summable sequence $\{q_k\}$ of real numbers. In the :Fourier domain, this relationship can be expressed as

$$\Psi(\omega) = \tfrac{1}{2} Q(e^{-j\omega/2}) \Phi(\frac{\omega}{2}) \tag{2.7}$$

Since we will often use the dilates and translates of the scaling function and wavelet, we will denote these as

$$\begin{aligned} \phi_{j,k}(x) &= \phi(2^j x - k) \\ \psi_{j,k}(x) &= \psi(2^j x - k) \end{aligned} \tag{2.8}$$

2.3 Dual Multiresolution Analysis

Associated with a multiresolution analysis, there is a dual multiresolution analysis

$$\cdots \subset \tilde{V}_0 \subset \tilde{V}_1 \subset \cdots$$

with a scaling function $\tilde{\phi}$ and wavelet $\tilde{\psi}$ Similar to (2.1, 2.6), the functions $\tilde{\phi}$ and $\tilde{\psi}$ obey the relationship

$$\begin{aligned} \tilde{\Phi}(\omega) &= \tfrac{1}{2} G(e^{-j\omega/2}) \tilde{\Phi}(\frac{\omega}{2}) \\ \tilde{\Psi}(\omega) &= \tfrac{1}{2} H(e^{-j\omega/2}) \tilde{\Phi}(\frac{\omega}{2}) \end{aligned} \tag{2.9}$$

for some G and H which satisfy

$$P(z)G^*(z) + Q(z)H^*(z) = 1 \tag{2.10}$$

$$P(z)G^*(-z) + Q(z)H^*(-z) = 0 \tag{2.11}$$

where "*" denotes complex conjugation.

As before, we will define the dilates and translates of the dual scaling function and dual wavelet as

$$\begin{aligned} \tilde{\phi}_{j,k}(x) &= \tilde{\phi}(2^j x - k) \\ \tilde{\psi}_{j,k}(x) &= \tilde{\psi}(2^j x - k) \end{aligned} \tag{2.12}$$

Any function $f \in \mathcal{L}^2(\mathbf{R})$ may be decomposed as

$$f(x) = \sum_{k=-\infty}^{\infty} \langle f,\ \tilde{\phi}_{0,k}\rangle \phi_{0,k} + \sum_{j=0}^{\infty}\sum_{k=-\infty}^{\infty} \langle f,\ \tilde{\psi}_{j,k}\rangle \psi_{j,k} \tag{2.13}$$

where $\langle \cdot,\ \cdot \rangle$ is the usual $\mathcal{L}^2$ inner product given by

$$\langle f,\ g\rangle = \int_{-\infty}^{\infty} f(x)g(x)dx,\ f, g \in \mathcal{L}^2(\mathbf{R})$$

The inner product of f with $\tilde{\psi}_{j,k}$ is called the wavelet coefficient associated with $\psi_{j,k}$. The decomposition in (2.13) of the function into its wavelet coefficient:: is known as the wavelet decomposition. It is the starting point for most image processing algorithms that are developed using multiresolution analysis.

In (2.13) the sums over j and k are infinite sums. In practice, however, the sums will be finite. We will be given data at a "finest resolution" **J.** Typically, images are 512 x 512 and we have $\mathbf{J} = 9$ because $2^9 = 512$. So the sum over j will be from 0 to $\mathbf{J} - 1$. Further, we will consider 2^j shifts of the wavelet at resolution j. That is, there will be 2^j values of k associated with resolution j. So k will vary from 0 to $2^j - 1$.

The dual multiresolution analysis is not necessarily unique. In fact, there may be an infinite number of choices for G and H. A table of various choices for the dual wavelets of spline-based multiresolution is given in [29, pg 277]. We may also consider a different family of multiresolution analyses called the "spline-variisnts". The filter

coefficients for these are given in [29, pg 279]. Another possiblity is to require that $G = P$ and $H = Q$, in which case we get orthogonal wavelets. If we do not impose this restriction, the resulting wavelets are called biorthogonal.

In our experiments, we have used four wavelets: Haar wavelet, spline of order 2, spline-variant and Daubechies D4. The Haar wavelet is the one given in [29, pg 277] for $N = N = 1$. The spline of order 2 is also given in [29, pg 277] and corresponds to $N = 9, N = 3$. The spline-variant is given in [29, pg 279] with $N = N = 4$. The Daubechies D4 can be found in [30, pg 129].

3. EMPIRICAL PROPERTIES OF WAVELET COEFFICIENTS

The goal of lossy wavelet-based image compression is to represent an image with just a few wavelet coefficients. Compression is achieved by having to encode just a few (as opposed to all) wavelet coefficients. In order to get a good representation of the image, we are primarily interested in those wavelet coefficients which are large in magnitude. Sorting of the coefficients in decreasing order of magnitude gives us the large coefficients first. An interesting fact we observe is that the wavelet coefficients, when so sorted, lie almost precisely on a curve of the form

$$y(x) = \frac{C}{(x + \triangle)^{\alpha}} \tag{3.1}$$

This is addressed in section 1 of this chapter.

Further compression will be obtained if it is possible to predict which coefficients may be large. To do this, we develop a top-down adaptive search (TAS) algorithm in section 2. A bound on the error in this prediction is developed in section 3. The error bound is relatively tight for images which are highly structured. In fact, this analysis yields a number (we call it y) which is indicative of the amount of structure in an image. We also derive a theoretical upper bound on the value of γ that will result when the TAS algorithm is applied to an autoregressive process. This theoretical bound further supports the empirical evidence that y is indicative of the amount of structure in the image. This property of y is discussed in section 4.

An important property of the function $y(x)$ in equation (3.1) is that it retains the same form when the argument x is scaled. This leads to the result that self-similarity may be observed in the wavelet domain even if no such property is apparent in the pixel domain. This is discussed in section 5.

An argument that has often been offered in support of wavelets is that they have the ability to zoom in on the details [29, 34]. However, the notion of "zooming in" has not been precisely defined. In section 6, we give a rigorous definition of this concept. Further, we demonstrate that the TAS algorithm automatically zooms in onto the edges in an image.

Although the TAS algorithm predicts scale-space positions and magnitudes, it does not predict the signs of the wavelet coefficients. In section 7, we present a simple scheme for predicting the signs.

The TAS algorithm may be used in conjunction with various data structures. The data structure we have found most successful is the one called the "web", which we define in section 8. All experiments in this thesis (except the one in section 3.9) use the TAS algorithm in conjunction with the web.

A data structure that is commonly used in wavelet-based image compression is the quadtree [3]. It is possible to use the TAS algorithm in conjunction with a quadtree. However, as shown in section 9, the TAS algorithm appears to work better with the web than with the quadtree.

An overview of the content of sections 1 through 7 of this chapter is shown in figure (3.1).

3.1 Coefficients ordered by magnitude

One way to compress an image is to retain only a few wavelet coefficients and set the remaining ones to zero. It has been observed that the visual quality of the compressed representation is not much affected when coefficients that are small in magnitude are set to zero. However, it is greatly affected when large coefficients are set to zero [3]. Since we would like this compressed representation. to bear a close resemblance to the original image, we would like to retain those coefficients which are large in magnitude.

We start by sorting the coefficients in decreasing order of magnitude. When this is done, the "best" representation using n coefficients will be obtained by retaining

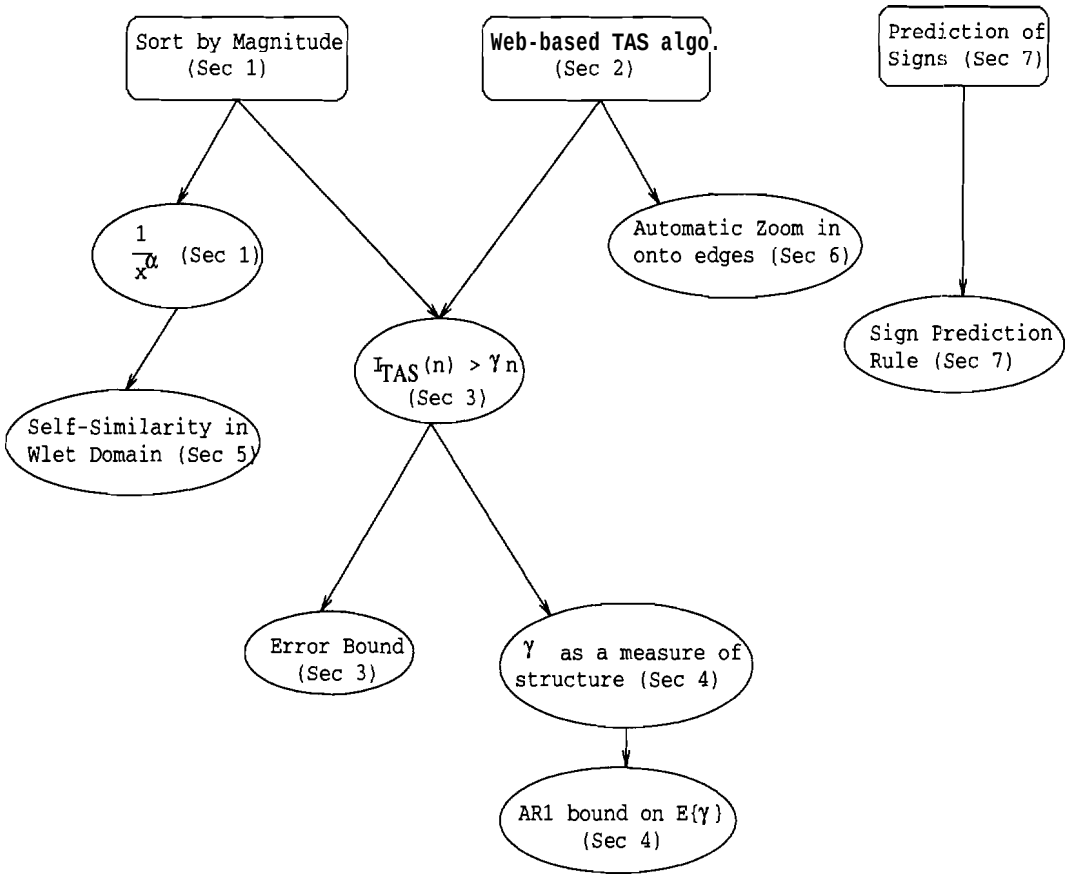


Fig. 3.1. Overview of this chapter: Relationship between various empirically observed properties.

the first n coefficients. More precisely, the minimum $\mathcal{L}^2$ error representation (when an orthonormal wavelet is used) is obtained by retaining the first n coefficients.

After sorting the coefficients by magnitude, we define a "Sort-Index" associated with each scale-space position as follows: The scale-space position with the nth largest coefficient-magnitude (of the N^2 coefficients computed for an N x N image) has a sort-index of n. For example, the scale-space position with the largest coefficient has a sort-index of 1. By construction, a plot of the magnitudes of the coefficients versus sort-index will be monotonically decreasing.

It can be seen from figure (3.2) that a plot of the coefficient-magnitudes versus sort-index obeys the form of equation (3.1). The data shown in this figure is for the spline-variant wavelet [2] applied to the Lenna image. In order to verify that equation (3.1) is the correct functional form for the data in figure (3.2), we take the logarithm on both sides of equation (3.1) and obtain

$$\log y = -\alpha \log(x + A) + \log C \tag{3.2}$$

This implies that a plot of logy versus $\log(x + A)$ should yield a straight line of slope $-\alpha$ and intercept log C. Figure (3.3) shows this plot. Typical values of α and C are $\alpha = 0.77$ and $C = 44000$ for the data ranging from x = 3 to 4000. The value of A must be found by a search method. A typical value is A = 9.

3.2 Coefficients chosen by Top-down Adaptive Search

We now look at a method for predicting which scale-space positions may have a large wavelet coefficient. The method is essentially a top-down adaptive search (TAS) algorithm that selects scale-space positions in a greedy manner. We will find that when the coefficients are plotted in the order in which this algorithm selects them, the plot looks very similar to the $\frac{1}{x^\alpha}$ plot of figure (3.2). We then conclude that the order in which the TAS algorithm generates coefficients is very close to the order in which they would appear when sorted by magnitude.

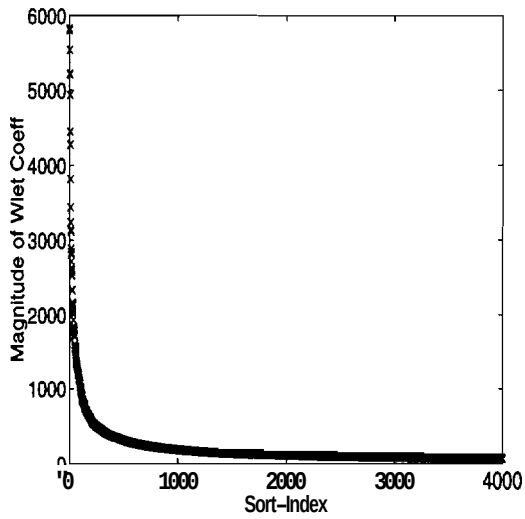


Fig. **3.2.** Plot of magnitudes of wavelet coefficients of the Lenna image after sorting by magnitude.

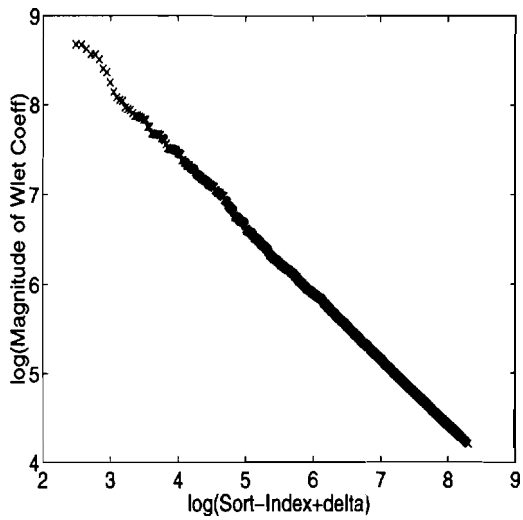


Fig. **3.3.** Log-log plot of magnitudes versus Sort-Index.

The TAS algorithm requires that the scale-space positions be indexed using a parent-child relationship. For 1-dimensional algorithms, the scale-space positions can naturally be indexed using a parent-child relationship [39, 40, 41]. These parent-child relationships define a data structure which places the coarsest scale-space position uppermost in the hierarchy. This idea has been extended for 2-dimensional wavelets using a quadtree [3]. However, we find that the TAS algorithm works better with the more generalized parent-child relationship called the "web" which we propose in section 3.8. Therefore, all experiments in this thesis (except the comparison in section 3.9) are performed using the TAS algorithm with the web.

The TAS algorithm starts by selecting the coarsest scale-space position. Then, at each stage, it selects one of the children of the previously selected scale-space positions. This selection is performed in a greedy adaptive manner as described below.

Let D_k be the set of k scale-space positions chosen at the first k steps. Let chil(D_k) be the children of the scale-space positions in D_k. The algorithm for choosing scale-space positions is as follows:

1. Let $k = 1$ and set $D_1 = \{$ The root node $\}$.
2. Let $d_k^* \in$ chil(D_k) be the scale-space position in chil(D_k) which has the largest (in magnitude) wavelet coefficient.
3. Set $D_{k+1} = D_k \bigcup d_k^*$, i.e. append d_k^* to D_k. Increment k.

Now iterate steps 2 and 3 until the desired number of scale-space positions has been chosen.

For each chosen scale-space position, we assign a "TAS-index" as follows: For the scale-space position chosen at the mth step, we assign a "TAS-index" of m. For the remainder of the thesis, "m" and "n" will respectively refer to the TAS-index and Sort-Index (which was defined in the beginning of section 3.1).

The TAS algorithm has the ability to go both up and down in resolution. In other words, the resolution of the node d_k (chosen at the kth iteration) may be greater than, equal to or less than that of d_{k+1}. This can be seen as follows. If a node d enters chil(D_k) at some iteration k, the d always remains in the list of children

chil(D_{k+1}), chil(D_{k+2}), $\cdots$, until d gets chosen by the algorithm. Since chil(D_k) is the set of possible candidates from which the algorithm may choose a node at the kth iteration, we can say: Once a candidate, always a candidate ... until chosen. Thus, nodes of various resolutions are present in the set of candidate nodes.

It is illustrative to view a plot of the magnitudes of the coefficients of the chosen wavelets versus the TAS-index. Each cross in figure (3.4) is a data point. The lower convex hull (i.e. the convex hull fitted from below the points) appears to closely resemble a curve of the form $\frac{1}{x^\alpha}$. We observe that most data points lie on a curve of the form $\frac{1}{x^\alpha}$ but a few points lie above the curve. Notice that practically none lie below the curve.

In figure (3.4), the coefficients that lie above the curve appear "too late" in the TAS ordering. For example, the 50th largest coefficient may appear at position 3500 on in the TAS algorithm. (That is, the coefficient has a Sort-Index of n = 50 and TAS-Index of m = 3500.) If a coefficient appears too late in the TAS ordering, then some other coefficient must appear too early. An interesting empirical observation is that we can put a bound on how early a coefficient may appear. This is discussed in the next section.

3.3 Comparison of TAS algorithm to Sorting

Figures (3.2, 3.4) indicate that the TAS algorithm is doing a fairly good job of capturing the large wavelet coefficients. We now quantify the effectiveness of prediction of the TAS algorithm. We find that we can put a bound on how early a coefficient may appear in the TAS ordering. In particular, we find that the TAS-Index must be greater than y times the Sort-Index, where γ is an image-dependent constant.

Since the TAS algorithm directly yields the TAS-Index of each scale-space position, we would like to use the TAS-Index to estimate the magnitude of the wavelet coefficients. Although we have a very accurate estimate based on the Sort-Index (equation (3.1)), we prefer to use the TAS-Index because it is directly available to

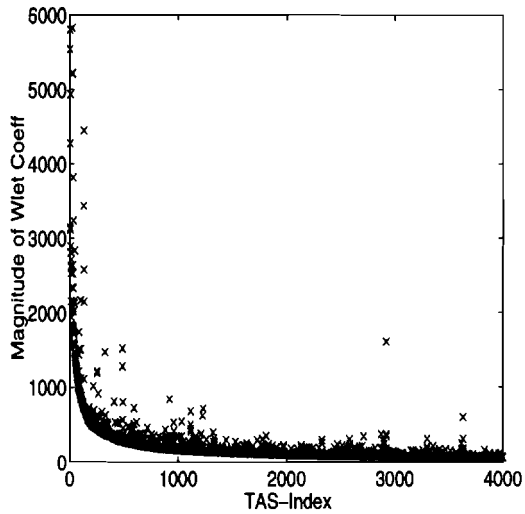


Fig. **3.4.** Plot of coefficients in the order generated by the TAS algorithm.

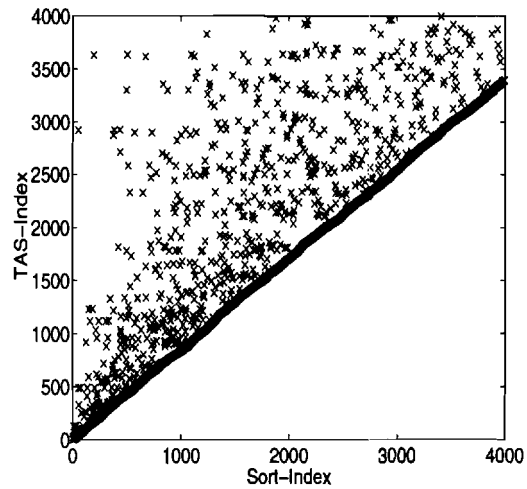


Fig. **3.5.** Plot of TAS-index versus Sort-Index.

us from the TAS algorithm whereas the Sort-Index is not. In this section, we will develop an estimate for the magnitudes based on the TAS-Index and also derive a bound on the error in this estimate.

3.3.1 TAS-Index versus Sort-Index

In section 3.1, we found that a plot of magnitudes versus Sort-Index lies on the curve given by equation (3.1). In section 3.2, we found that a plot of magnitudes versus TAS-Index also lies on a similar curve. This suggests that; the TAS-index closely matches the sort-index. To verify this, we plot the TAS-index versus the Sort-Index in figure (3.5).

In order to understand figure (3.5), we define I_{TAS} (n) to be the TAS-index of the nth largest wavelet coefficient (i.e. the wavelet coefficient whose sort-index is n). It is clear from figure (3.5) that there exists a number γ such that

$$I_{TAS}(n) \geq \gamma\, n \tag{3.3}$$

The value of γ is image dependent and it typically ranges between 0.8 and 0.9.

The following argument shows that the permissible range of y is $0 \leq \gamma \leq 1$. Since n and I_{TAS} (n) are both positive integers, it is clear from equation (3.3) that $\gamma \geq 0$. Further, since $n \geq 1$ and there is a value of n for which I_{TAS} (n) = *1*, it follows that $\gamma \leq 1$. The range of γ is therefore given by $0 \leq \gamma \leq 1$.

3.3.2 Estimation of Magnitude using TAS-Index

For the purposes of image compression, it is of interest to be able to approximate the magnitudes of the coefficients using the TAS-index. We already have such an approximation based on the Sort-Index: From equation (3.1) it follows that the magnitude of the nth largest coefficient can be approximated by

$$\frac{C}{(n+\triangle)^{\alpha}} \tag{3.4}$$

A simple approximation based on the TAS-index is to replace n by I_{TAS} (n) in the above expression. The magnitude of the nth largest coefficient is then approximated

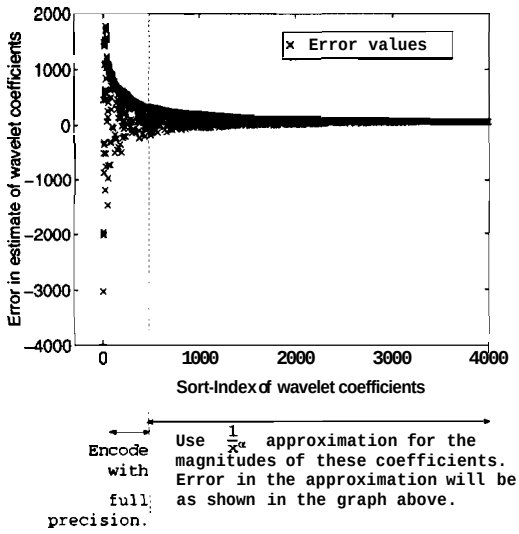


Fig. *3.6.* Error in our estimate of magnitude.

by

$$\frac{C}{(I_{TAS}\,(n)+\triangle)^{\alpha}}$$

The error in this approximation can be written as

$$\varepsilon(n) \;=\; \frac{C}{(I_{TAS}\,(n)+\triangle)^{\alpha}} - \frac{C}{(n+\triangle)^{\alpha}} \tag{3.5}$$

From equation *(3.3)* and the fact that $I_{TAS}\,(n) < \infty$, we get

$$-\frac{C}{(n+\triangle)^{\alpha}} \;\leq \varepsilon(n) \;\leq (\frac{1}{\gamma^{\alpha}}-1)\frac{C}{(n+\triangle)^{\alpha}}, \quad \forall n > \triangle \tag{3.6}$$

Thus, equation *(3.6)* gives us a bound on the error in our approximation. The error in the approximation decreases as $\frac{1}{n^{\alpha}}$. **A** plot of this error is shown in figure *(3.6).* Typically, past the knee of the $\frac{1}{x^{\alpha}}$ curve, the approximation becomes useful. However, the error bound is loose for small values of n and this approximation does not appear to be useful for the largest few nodes. Consequently, the image compression algorithm presented in chapter *4* uses a form of differential pulse code modulation [27, pg 484] for the first few values of *n* and then uses the approximation of equation *(3.4)* for large values of n.

The TAS algorithm is a ***local*** optimization method whereas the Sort-Index is obtained by a global optimization scheme. Therefore, the number of scale-space positions considered by the TAS algorithm is less than that for a global search. For example, to find the optimal scale-space position at the 1000th step, the TAS algorithm may consider only 4000 possible candidate scale-space positions, whereas a global search would consider $512' = 262144$. To encode one of K possibile outcomes, we require $\log_2(K)$ information bits. Consequently, it requires a smaller number of bits to encode the scale-space positions based on their TAS-index. If only the TAS-indices are available to the decoder, it is necessary to be able the approximate the magnitudes based on the TAS-index. This is the reason why we developed the above approximation.

As a final point, we note that it is not of much concern that equation (3.6) does not hold for $n \leq A$. The reason is as follows. A typical value of $\triangle$ is $A = 9$ and a typical range for n is 1 to 64000. Thus, $n \leq A$ represents only the first few values of n and in any case, the bound is fairly loose for low values of n. The knee of the curve of figure (3.2) occurs around $n = 500$, and we use the approximation only for $n > 500$.

3.4 y as a Measure of Image Structure

We now show that this number y (of equation (3.3)) is indicative of the amount of structure present in the image. Images for which the value of y is high tend to be highly structured, and vice versa. This indicates that for structured images, the TAS algorithm does a good job of ensuring that small coefficients appear late in the TAS ordering.

In order to theoretically demonstrate that y is indicative of the .amount of structure in an image, we do the following. We consider the first order autoregressive (AR) process for which it is well known that the AR parameter "***a***" is indicative of the amount of structure in the process. We show that when the TAS algorithm is applied to this process, an upper bound on the expected value of y decreases as "***a***"

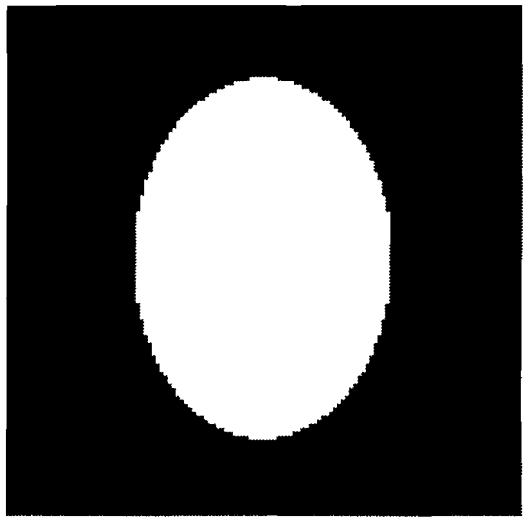


Fig. 3.7. Image of an oval blob.

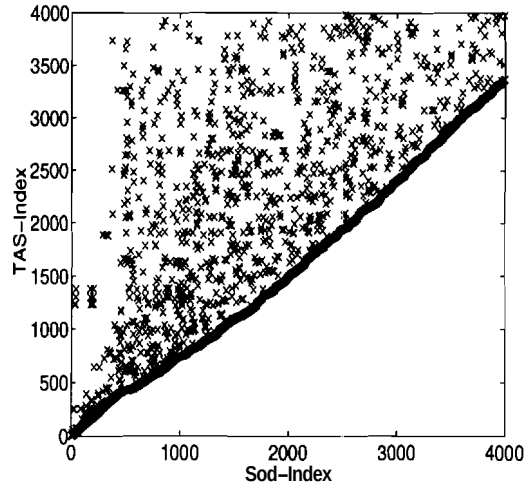


Fig. 3.8. $\gamma = 0.75$ for the oval blob.

decreases. This upper bound depends on the maximum eigenvalue of an associated linear transformation.

3.4.1 Empirical Observations

We find that for images that are very unstructured (i.e. look like noise), the value of γ is somewhat lower than the typical value of 0.8. This is shown in figure (3.7-3.12) where we show (i) an image containing an oval blob (y = 0.75), (ii) the same image with noise added to the background ($\gamma = 0.08$) and (iii) pure white noise (y = 0.0). As can be seen from the figures alongside, the value of γ decreases as the image becomes more unstructured.

3.4.2 Bound on $E\{\gamma\}$ of an AR1 process

We will now give a theoretical interpretation of the fact that as an image becomes more unstructured, the value of γ decreases towards zero. We do this by applying the TAS algorithm to a first order autoregressive (AR1) process.

The key result of this subsection is the following: An upper bound on $E\{\gamma\}$ decreases as the AR1 parameter a decreases. This is shown in figure (3.13).

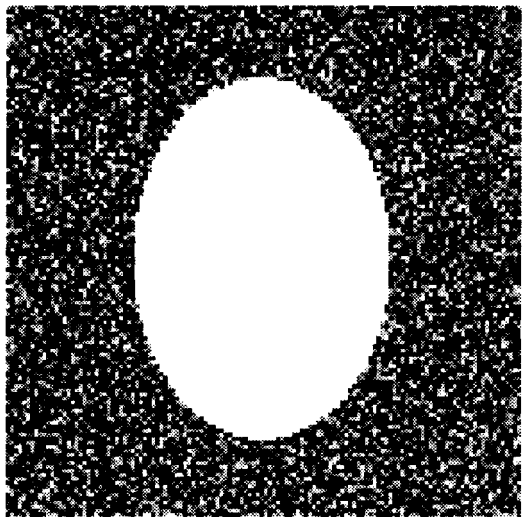


Fig.3.9. Imageofanovalblobwith background noise.

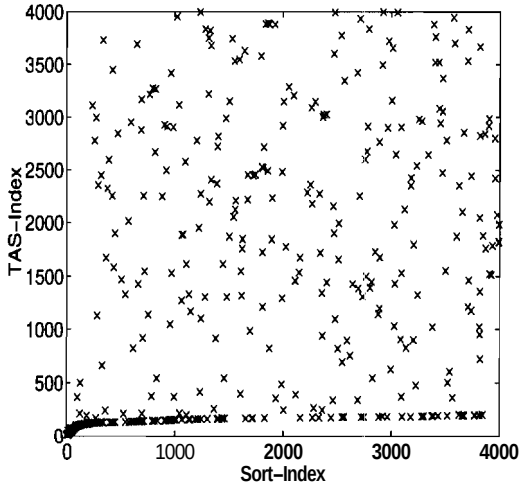


Fig.3.10. $\gamma = 0.08$ for the oval blob with background noise.

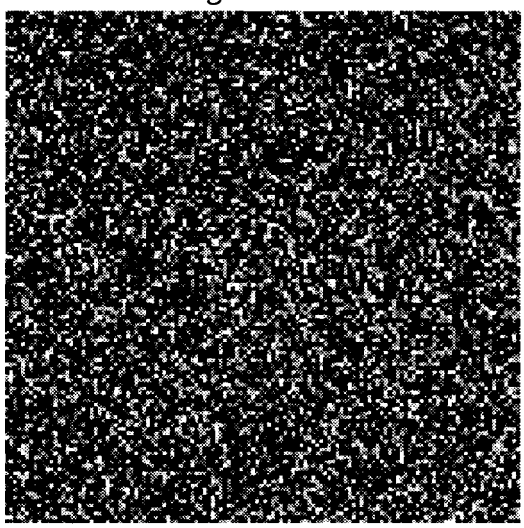


Fig. 3.11. Uniformly distributed random noise.

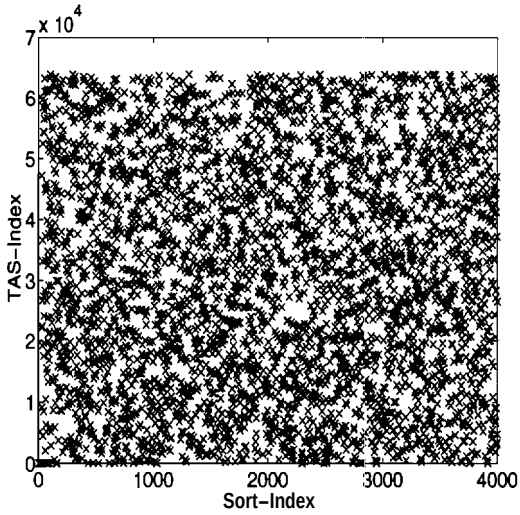


Fig. 3.12. $\gamma = 0.0$ for the image of uniform random noise.

Consider a first-order autoregressive (AR1) process

$$y_n = ay_{n-1} + \xi_n, \quad n = 1, \cdots, N \tag{3.7}$$

where $0 \leq a \leq 1$ and the ξ_n's are independent, identically distributed zero-mean Gaussian random variables with variance $\boldsymbol{a^2}$. If a 1-dimensional version of the TAS algorithm is applied to this AR1 process, the resulting value of y is a random variable which depends on the AR parameter a. Recall that small values of a correspond to a process that has less correlation and is therefore more unstructured.

To compute an upper bound on γ, we first rewrite equation (3.7) as

$$\mathbf{y} = \mathbf{A}\xi$$

where $y = [y_1, \cdots, y_N]^T, \xi = [\xi_1, \cdots, \xi_N]^T$, and A is an N x N matrix which is a function of the AR1 parameter "a". Now the discrete-time wavelet transform of y may be written as

$$\mathbf{c} = \mathbf{Hy}$$

where $\boldsymbol{c} = [c_1, \cdots c_N]^T$ is a vector of wavelet coefficients and H is an orthonormal N x N matrix.

Since the ξ_i's are zero-mean Gaussian, so is $\boldsymbol{c}$. Further,

$$E\{\mathbf{cc}^T\} = \sigma^2 \mathbf{HAA}^T\mathbf{H}^T \tag{3.8}$$

where $\boldsymbol{a^2}$ is the variance of any ξ_i. We can now find a bound on $E\{\gamma\}$ for the AR1 process.

Theorem 1 : Let $\lambda_{\max}(a)$ be the largest eigenvalue of $\mathbf{HAA}^T\mathbf{H}^T$. For the AR1 process of equation (3.7), we have

$$E\{\gamma\} \leq \frac{\sqrt{\lambda_{\max}(a)}}{N} \sum_{n=1}^{N} \frac{1}{n} \tag{3.9}$$

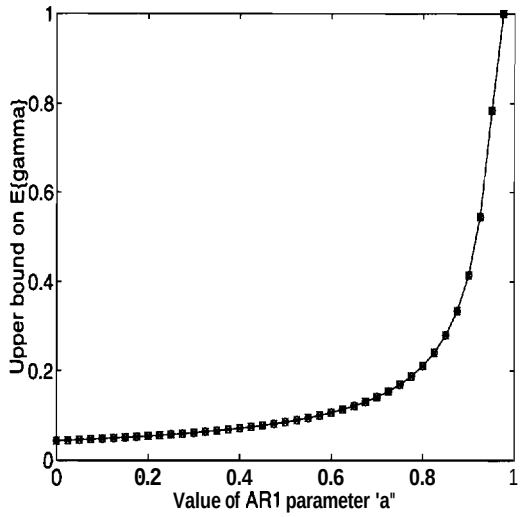


Fig. 3.13. Bound on $E\{\gamma\}$ versus AR1 parameter "a".

Proof: The proof is in the appendix. □

Note that in the above bound, $\lambda_{\max}$ is a function of the AR1 parameter "a". A plot of the bound versus a is shown in figure (3.13). It can be seen that this upper bound decreases as the AR parameter a decreases. This offers a partial explanation for the behaviour of γ in figures (3.7 - 3.12) where it is seen that as the image becomes more unstructured, the value of γ decreases towards zero. Note that from the discussion of section 3.3.1, it follows that $E\{\gamma\} \leq 1$. So the data plotted in the graph of figure (3.13) is the minimum of 1 and the bound of equation (3.9).

3.5 Self-similarity in the Wavelet Domain

While fractal images like the maple-leaf or fern-leaf [42, figures 2.4, 2.5] exhibit an obvious self-similarity, such self-similar behaviour is not apparent in an image like the Lenna image. In this subsection we will find that although a subimage may not look like the original, there is a form of self-similarity present in the ordered wavelet coefficients. Thus, although there may be no self-similarity in the pixel domain, one can find a self-similarity in the wavelet domain for certain subimages.

Consider a subimage of the original image as indicated by the box around Lenna's right eye in figure (3.14a). The wavelets corresponding to certain scale-space positions

lie entirely within this box. Since each scale-space position is associated with a sort-index, we may say that the wavelets associated with certain sort-indices lie entirely within this box. We construct a binary vector v as follows: The nth position of v contains a "1" if the wavelet associated with the sort-index of n hiss support inside the box; and a "0" otherwise More precisely, let

$$\mathbf{v} = [v_1, v_2, \cdots, v_{N^2}]^T$$

for an image of size N x N. Then each v_n is given by

$$v_n = \begin{cases} 1 & \text{if support}(\psi_{d_n}) \subset \text{Box} \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

Thus v is a "subsampling vector" which indicates which of the sort-indices need to be subsampled to construct the subimage inside the box.

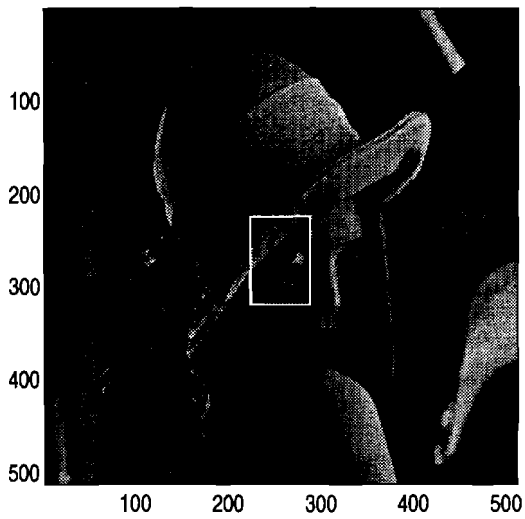
The definition of v_n in equation (3.10) assumes that the Haar wavelet is used. For other wavelets, the nth position of v has a "1" if the *dyadic rational pair* (defined in section 3.6) associated with the scale-space position whose sort-index is n lies inside the box. Approximately speaking, the center of the support of the nth wavelet has to lie inside the box for v_n to be 1.

If the subsampling is close to uniform, then a plot of the cumulative sum of v will appear like a straight line. Further, we could find the magnitudes of these wavelet coefficients by substituting $k_{\text{sub}}x$ for x in equation (3.1). The resulting curve for the coefficients of the subimage is given by

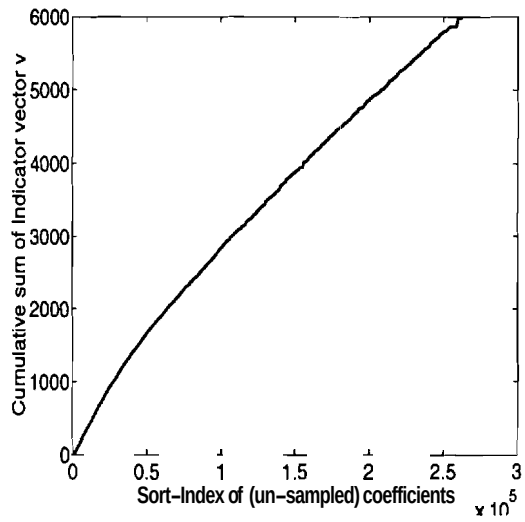
$$y_{\text{sub}} = \frac{C}{(k_{\text{sub}}x + \triangle)^\alpha} = \frac{C/k_{\text{sub}}^\alpha}{(x + \triangle/k_{\text{sub}})^\alpha} \tag{3.11}$$

It is easy to see that this is exactly the same form as equation (3.1) with C and A replaced by C/k_{sub}^α and $\triangle/k_{\text{sub}}$ respectively. The value of α is unchitnged.

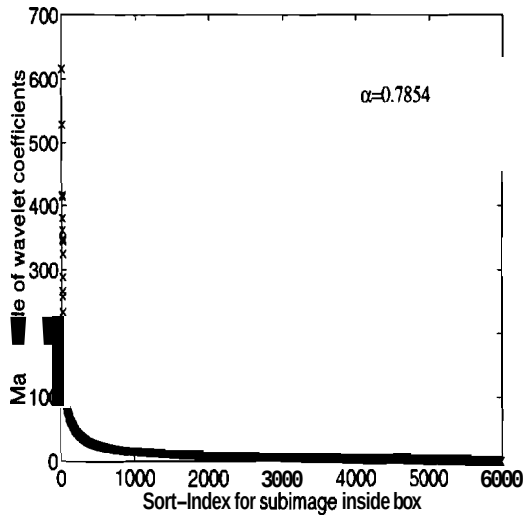
A plot of the cumulative sum of the subsampling vector v is shown in figure (3.14b). What we find is that the plot is not quite a straight line. Yet, we find that the sorted coefficients of the subimage do indeed lie on a curve of the form $\frac{1}{x^\alpha}$



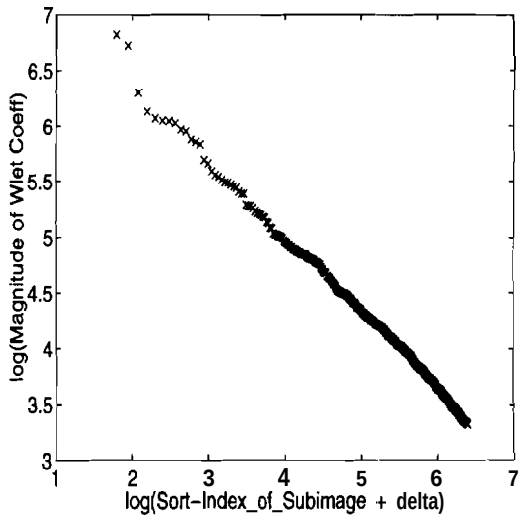
(a) Subimage is inside box.



(b) 5985 wavelets contribute to box in (a).

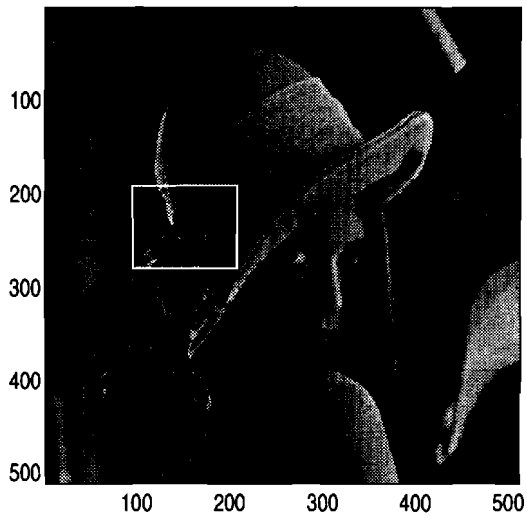


(c) Magnitudes of coeffs computed for subimage in (a).

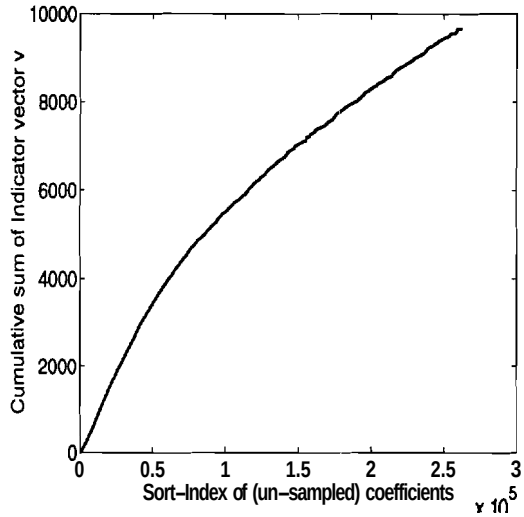


(d) log-log plot of (c).

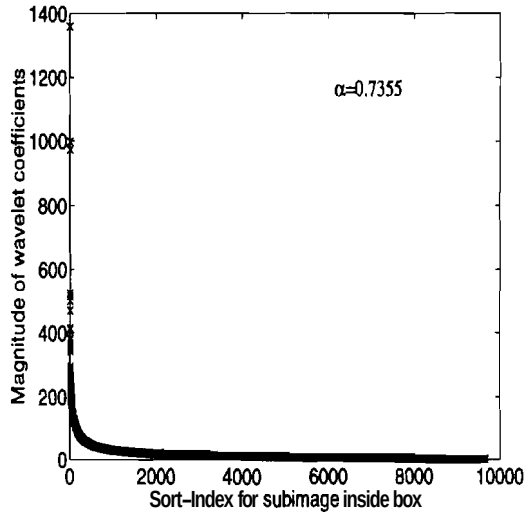
Fig. 3.14. Subimage: The wavelet coefficients of a subimage also obey the $\frac{1}{x^{\alpha}}$ curve. Figures (a,e,i) shows the subimage. Figures (b,f,j) show the cumulative sum of subsampling vector v. Figures (c,g,k) show magnitudes of wavelet coefficients computed for subimage and their log-log plot are respectively in (d,h,l). The *a* of the original image is 0.7450.



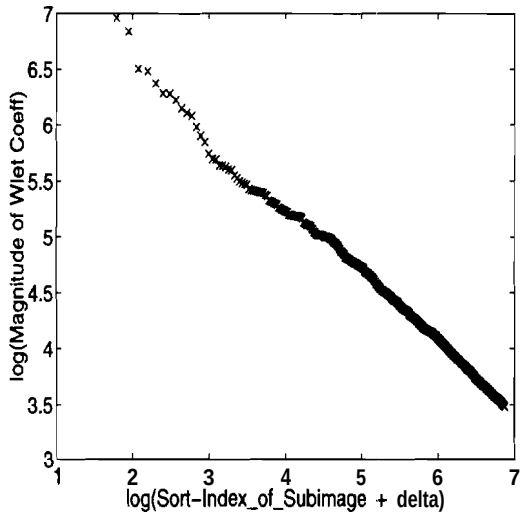
(e) Subimage is inside box.



(f) 9657 wavelets contribute to box in (a).

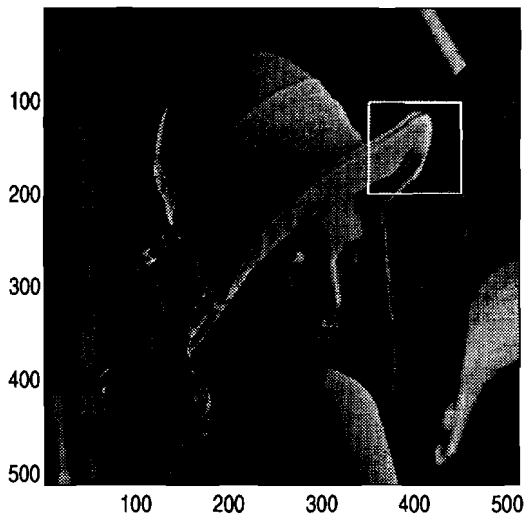


(g) Magnitudes of coeffs computed for subimage in (a).

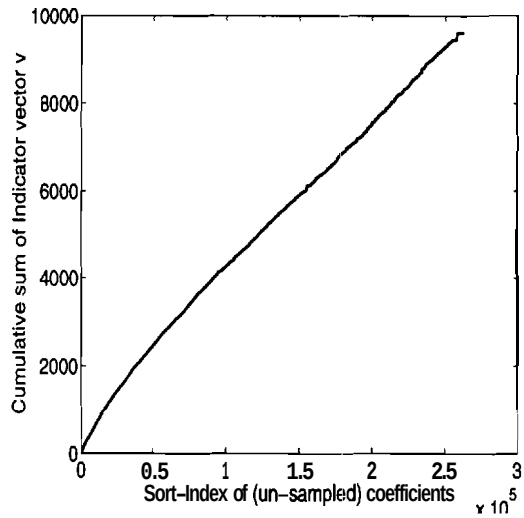


(h) log-log plot of (c).

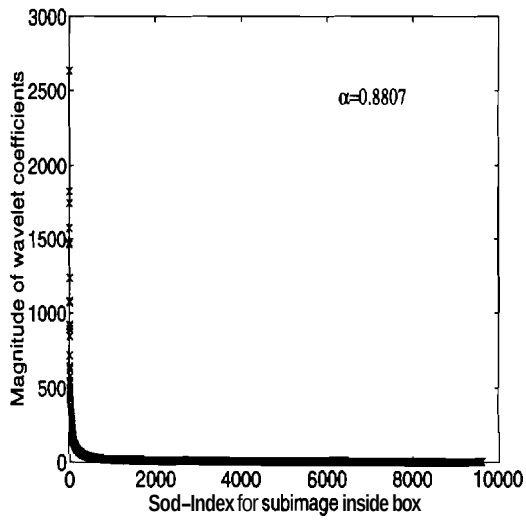
Fig. 3.14. Subimage: Continued



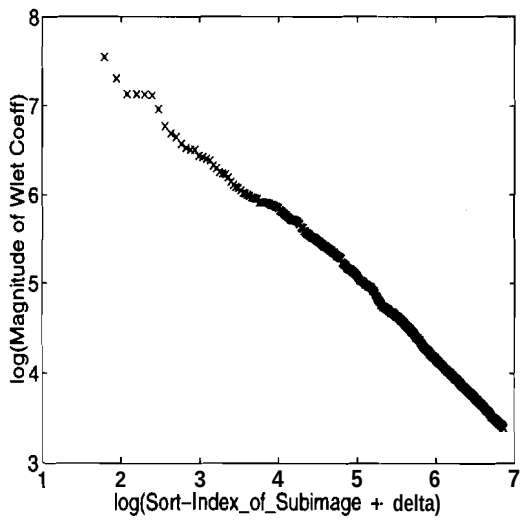
(i) Subimage is inside box.



(j) 9603 wavelets contribute to box in (a).



(k) Magnitudes of coeffs computed for subimage in (a).



(I) log-log plot of (c).

Fig. 3.14. Subimage: Continued

(see figures (3.14c,d)). The value of a is close to that of the original image. Other examples are given in figures (3.14e,f,g,h) and (3.14i,j,k,l).

Various researchers [43, 44, 45] have explored the connection between wavelets and $\frac{1}{f}$ processes (i.e. stochastic processes whose power spectral density falls off as $\frac{1}{f^\alpha}$). In fact, the wavelet transform is used as a whitening filter for a $\frac{1}{f}$ process [43]. The similarity between these works and ours is the focus on the interplay between wavelets and a behaviour of the form $\frac{1}{x^\alpha}$ of the given signal. What distinguislhes our approach from theirs is that we observe the $\frac{1}{x^\alpha}$ behaviour along a particular l-dimensional path in 2-dimensional scale-space. In contrast, the approaches of [43, 44, 45] directly consider a l-dimensional signal. Therefore, we add to their work by making the following contribution: We show that even for a 2-dimensional signal, it is possible to observe a $\frac{1}{x^\alpha}$ behaviour by appropriately selecting a path in scale-space. In other words, the sequence of scale-space positions (corresponding to sort-indices of 1, 2, 3, ..) defines a path in 2-dimensional scale-space. It is along this path that we are able to observe the $\frac{1}{x^\alpha}$ behaviour.

An interesting connection between our approach and fractal-based techniques is that fractal-based image compression techniques rely on self-similarity in the pixel domain [46, 47, 48]. Another connection comes from the work of Davis [49], who shows that the fractal approach itself can be subsumed within the wavelet framework. It is therefore possible that there exists a deeper connection between our wavelet-based image analysis and the various fractal-based approaches for image compression.

3.6 Automatic Zooming-in onto Edges

In this subsection, we state a theorem which asserts that the TAS algorithm zooms in at a point in the image only if there exists an edge passing through that point. We also present some experimental results that support the statement of the theorem and examine some uses for this property.

In order to precisely define what "zooming in" means, we examine how scale-space positions are indexed. So far, we have been using the letter d (as in equation (1.1))

to indicate a 2-dimensional scale-space position. Associated with d, there is a dilated and translated version ψ_d of the original l-dimensional wavelet ψ. We write ψ_d as

$$\psi_d(x,y) \;=\; \psi(2^{j_x}x - k_x)\cdot\psi(2^{j_y}y - k_y) \tag{3.12}$$

We therefore define the 2-dimensional scale-space position d to be $d = (d_x, d_y)$, where d_x and d_y are l-dimensional scale-space positions which are written as

$$d_x = \frac{2k_x+1}{2^{j_x}}, \qquad d_y = \frac{2k_y+1}{2^{j_y}} \tag{3.13}$$

Given a scale-space position d, we can now refer to the x resolution and y resolution of d as $j_x(d)$ and $j_y(d)$ respectively. Large values of j_x and j_y denote scale space positions with high resolution in x and y directions respectively.

Since j_x, j_y k_x, k_y are non-negative integers, it follows that d_x, d_y are dyadic rationals. Further, we are interested only in those values of k_x, k_y that satisfy $0 \leq k_x < 2^{j_x}$ and $0 \leq k_y < 2^{j_y}$. This is because we may assume, without loss of generality, that the 2-dimensional image is supported on the unit square $(0,1] \times (0,1]$. We are therefore interested in those 2-dimensional scale-space positions $d = (d_x, d_y)$ for which d_x and d_y are dyadic rationals in $(0,1]$.

We can now say that a sequence of scale-space positions (i.e. dyadic rationals) zooms in at a point p if we can find a subsequence converging to p which has the property that the support of the associated wavelets decreases to zero in both x and y. Formally, this may be written as:

Definition: A sequence of scale-space positions $\{d_m\}_{m\geq 1}$ is said to zoom in at a point $p \in (0,1]$ x $(0,1]$ if there exists a subsequence $\{d_{m_k}\}_{k\geq 1}$ that converges to p which has $j_x(d_{m_k})$ and $j_y(d_{m_k}) \to \infty$ as $k \to \infty$.

We now need to rigorously define what an "edge" is. To do this, we first let $g(x,y)$ be a function composed of piecewise polynomials. We assume that there are L pieces, and $\mathcal{I}_l$ is the indicator function of the lth piece. The function $g(x,y)$ can be written

$$g(x,y) \;=\; \sum_{l=1}^{L}\sum_{p=1}^{M}\sum_{q=1}^{M} a_{pq}^{(l)}\, x^p\, y^q\, \mathcal{I}_l(x,y) \tag{3.14}$$

For the above piecewise polynomial function, we define an edge as follows:

Definition: An edge of $g(x, y)$ is the boundary of any region $\mathcal{I}_l(x, y)$.

We are now ready to state a theorem that shows a zooming in property of the TAS algorithm.

Theorem 2 : Let $\{d_m\}_{m \geq 1}$ be the sequence of nodes selected by the TAS algorithm when used with a compactly supported wavelet of regularity M and applied to the image $g(x,y)$ of equation (3.14). Then the TAS algorithm zooms in onto a point p $\in (0, 1]$ x $(0, 1]$ only if there exists an edge passing through p.

Proof: The proof is in the appendix. □

In practice, we have only a finite number of data samples which we interpolate in some manner to reconstruct what we believe closely resembles the original image. One method of interpolation is using splines [50], which results in a piecewise polynomial reconstruction $g(x, y)$ of the form (3.14). This function $g(x, y)$ can now be analysed using the TAS algorithm. The theorem guarantees that the TAS algorithm will zoom in on the edges of the $g(x,y)$.

It is important to note that an infinite subsequence may be required to converge to a point on an edge. As an example, consider the image which is 1 if x $< \frac{1}{3}$ and 0 otherwise. This image has only one edge. When this is analysecl using the Haar wavelet, we get an infinite sequence of scale-space positions converging to the edge at x $= \frac{1}{3}$. This is because the number $\frac{1}{3}$ is not a dyadic rational and hence the image can never be exactly represented using a finite number of Haar wavelets.

Experimental results that illustrate the theorem are shown in figures (3.15, 3.16) where we have placed a black dot to indicate each chosen scale-space position. There are a total of 15000 chosen positions shown (i.e. 15000 black dots) out of a total possible of 512 x 512 = 262144 scale-space positions. The chosen nodes are shown together with the original image. It is easy to see that there is a high concentration of chosen scale-space positions near the edges. This dense concentration occurs, for

example, near the outline of the face, shoulder, etc. It also occurs in regions with many small edges such as the hair and mustache.

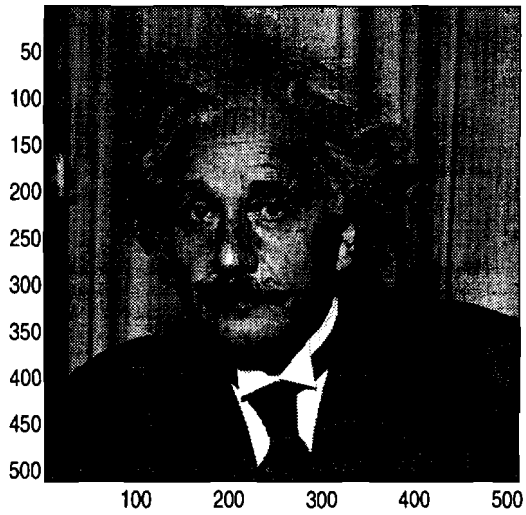
The utility of theorem 2 lies in its ability to authenticate candidate edges generated by certain edge detection algorithms [51, 52]. Many edge detection algorithms rely on ***local*** information to generate candidate edges. On the other hand, the TAS algorithm chooses scale-space positions based on wavelet coefficients which are computed using ***global*** image information. In practice, theorem 2 can be used to authenticate a candidate edge by seeing if a large number of scale-space positions lie within an ϵ distance of the candidate edge.

3.7 Prediction of Signs

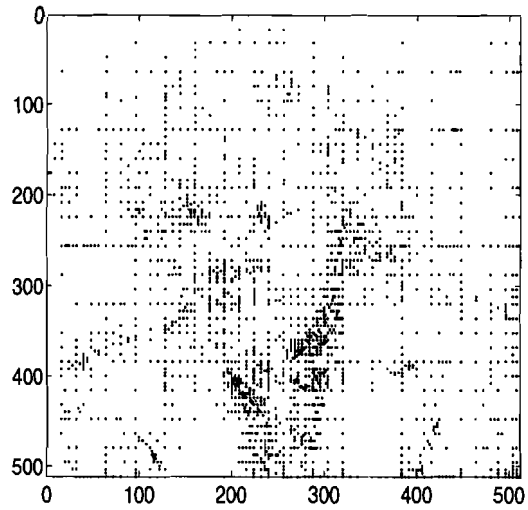
In this section, we address the issue of predicting the signs of the wavelet coefficients. The method outlined here can be used in conjunction with many image compression algorithms (such as the ones in [53, 3]). This will result in a modest improvement in bitrate for a fixed image quality. So far, we have not come across any image compression algorithm in the literature that specifically tries to predict the signs of the wavelet coefficients.

In order to predict the signs, it is necessary to identify some quantifiable behaviour of the reconstructed image that is sensitive to the signs. The behaviour that we use is based on the observation (presented in chapter 4) that an oscillatory artifact results when the magnitudes of the coefficients are encoded with low predision. The effect should be more pronounced when the sign of the coefficient is flipped because that results in a large error in the magnitude of the coefficient.

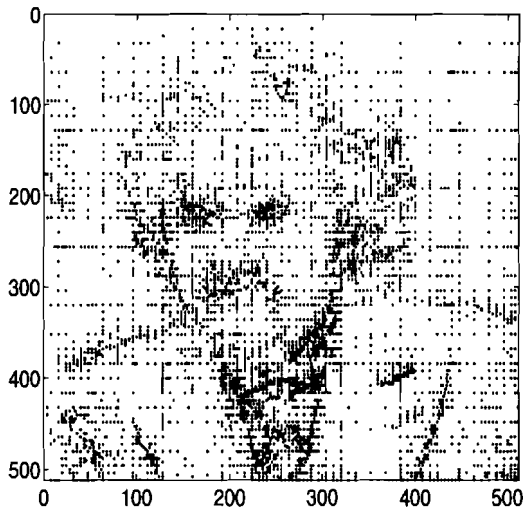
In figure (3.17)) we show a simple 1-dimensional example that illustrates the oscillatory artifact caused by using an incorrect sign. Figure (3.17a) shows the original which contains 512 data points. Figure (3.17b) shows the reconstruction using the 50 largest wavelet coefficients. The coefficients in figure (3.17b) have full precision. Figure (3.17c) shows the reconstructed function when one coefficient is used with an



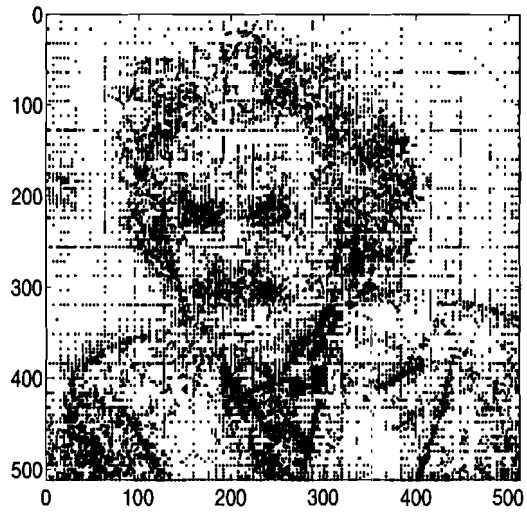
(a) Einstein.



(b) First 2000 scale-space positions chosen.

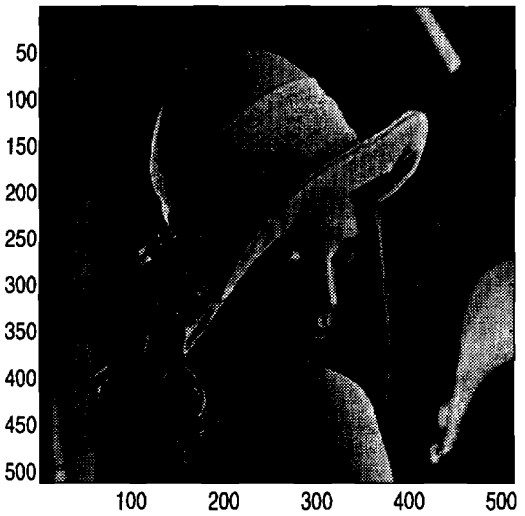


(c) First 5000 scale-space positions chosen.

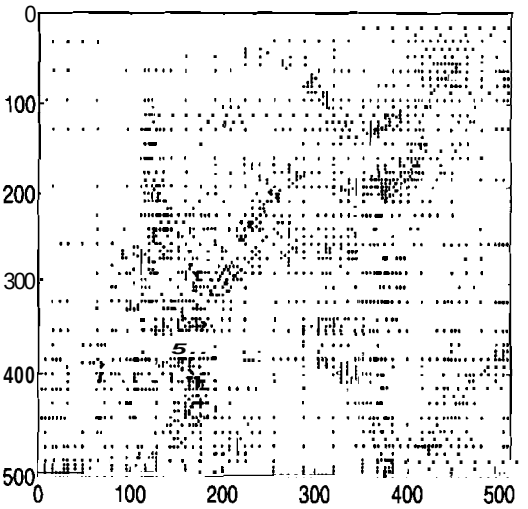


(d) First 15000 scale-space positions chosen.

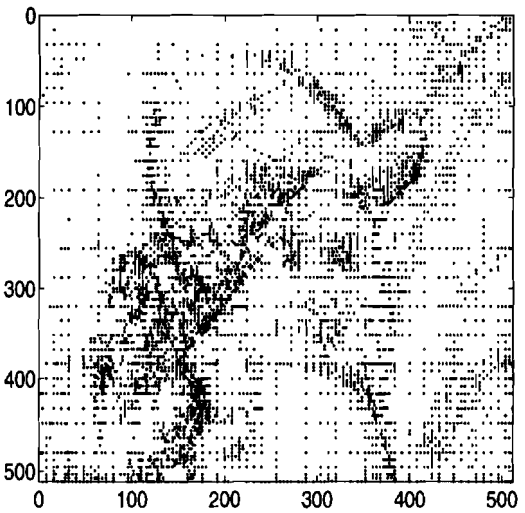
Fig. 3.15. Einstein image: The scale-space positions chosen by the TAS algorithm automatically zoom in on the edges. The chosen scale-spisce positions are indicated with a black dot. We have shown the first 2000, 5000 and 15000 chosen scale-space positions out of a total of 512 x 512 = 262144 positions.



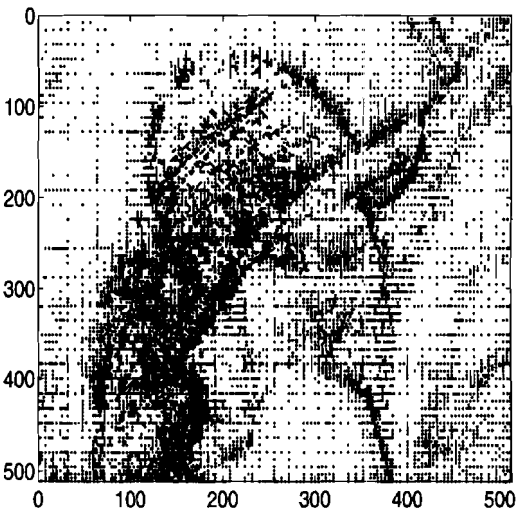
(a) Lenna.



(b) First 2000 scale-space positions chosen.



(c) First 5000 scale-space positions chosen.



(d) First 15000 scale-space positions chosen.

Fig. 3.16. Lenna image: The scale-space positions chosen by the TAS algorithm automatically zoom in on the edges. The chosen scale-space positions are indicated with a black dot. We have shown the first 2000, 5000 and 15000 chosen scale-space positions out of a total of $512 \times 512 = 262144$ positions.

incorrect sign. Notice that in the region near x = 100, a significant oscillatory artifact is introduced.

This property can be used to formulate a quantifiable measure to predict the sign of the wavelet coefficients. Let $d_1, d_2, d_3, \ldots$ be scale-space positions and let

$$f_{k-1}(p,q) = \sum_{i=1}^{k-1} c_{d_i} \psi_{d_i}(p,q) \tag{3.15}$$

be an approximation of the image f using k – 1 wavelet coefficients. The variables p, q above represent pixel locations. Further, let

$$\begin{aligned} f_k^+ &= f_{k-1} + |c_{d_k}| \psi_{d_k} \\ f_k^- &= f_{k-1} - |c_{d_k}| \psi_{d_k} \end{aligned} \tag{3.16}$$

We now have two candidate approximations f_k^+ and f_k^-. In order to measure the oscillations present in these two candidate approximations, let

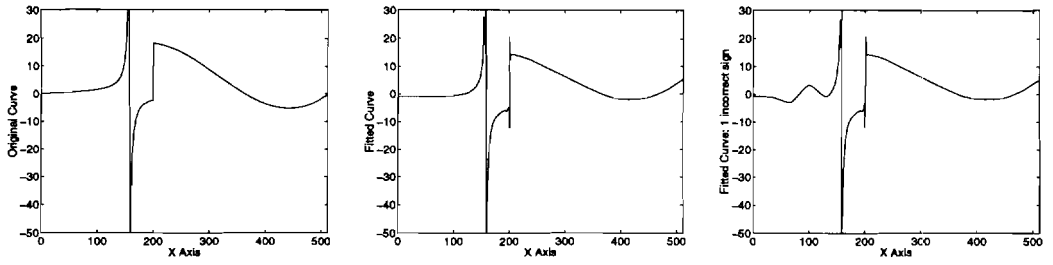
$$\begin{aligned} s_k^+ &= \sum_p \sum_q |f_k^+(p+1, q+1) - f_k^+(p,q)| \\ s_k^- &= \sum_p \sum_q |f_k^-(p+1, q+1) - f_k^-(p,q)| \end{aligned} \tag{3.17}$$

where the sum is over the pixels in the image. In the above equations, we are estimating the amount of oscillations by summing the absolute values of differences in the function value between adjacent pixels.

Our choice is between f^+ and f^- and we would like to pick the one that has less oscillatory behaviour. So our prediction of the sign of the kth coefficient is based on the following rule:

$$\begin{aligned} &\text{if} \quad s_k^+ < s_k^-, \quad \text{then set } f_k = f_k^+ \\ &\text{if} \quad s_k^+ \geq s_k^-, \quad \text{then set } f_k = f_k^- \end{aligned} \tag{3.18}$$

The results of using this method to predict the signs are shown in table (3.1). For each image, the TAS algorithm was run for 20000 coefficients. Each entry of table



(a) Original 1-dimensional curve.

(b) Fitted curve using largest 50 wavelet coefficients.

(c) Fitted curve with the sign of one wavelet coefficient flipped.

Fig. 3.17. Incorrect sign causes wiggles in the fitted curve (near x = 100)

(3.1) shows the accuracy in predicting the signs of these 20000 coefficients. To put these numbers in perspective, note that a accuracy of 50% can be obtained simply by flipping a fair coin. Our method yields an accuracy of around 70%. This is still somewhat far from the ideal goal of 100%. However, when this method is used in conjunction with the compression method of [53], it yields a modest reduction (about 8%) in the bit-rate.

3.8 Definition of a Web

We now give a precise definition of the notion of "children" that has been used to formulate the TAS algorithm. The data structure resulting from this definition of children is named a "web".

In order to define the children of a 2-dimensional scale-space position d, we first express d as an ordered pair $d = (d_x, d_y)$ of two 1-dimensional scale-space positions d_x and d_y. As mentioned in equation (3.13), the 1-dimensional scale-space positions are dyadic rationals in the unit interval $(0, 1]$, and are given by

$$d_x = \frac{2k_x + 1}{2^{j_x}}, \qquad d_y = \frac{2k_y + 1}{2^{j_y}}$$

The integers j_x and j_y are the x and y resolution of the scale-space position d. The integers k_x and k_y are the x and y shift that constitute d.

Image	D4 wavelet	Spline wavelet	Spline-variant
Car	71.150%	74.725%	69.900%
Couple	70.175%	72.675%	70.200%
Einstein	76.175%	74.875%	75.125%
F16	75.875%	75.375%	73.500%
Lenna	76.700%	75.075%	77.975%
Peppers	78.850%	77.575%	82.375%
Sailboat	72.125%	74.525%	72.500%
Stream	68.675%	72.250%	66.725%
Bank	77.375%	77.450%	78.900%
Widget	80.900%	78.475%	81.425%

Table 3.1 Percentage accuracy in prediction of signs for 10 images.

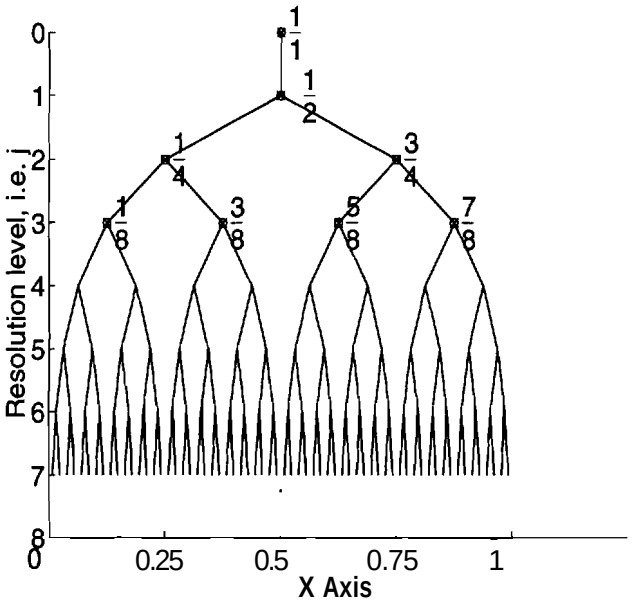


Fig. 3.18. There is a one to one correspondence between nodes on the binary tree and dyadic rationals in $(0,1]$. The dyadic rational corresponding to a node is simply the x coordinate of the node in the above diagram.

The children of the 1-dimensional nodes d_{x} and d_{y} can be expressed as follows.

$$\mathrm{chil}(d_{\mathrm{x}}) = \{d_{\mathrm{x}_L}, d_{\mathrm{x}_R}\}, \quad \mathrm{chil}(d_{\mathrm{y}}) = \{d_{\mathrm{y}_L}, d_{\mathrm{y}_R}\} \tag{3.19}$$

where

$$\begin{aligned} d_{\mathrm{x}_L} &= \frac{2(2k_{\mathrm{x}}+1)-1}{2^{j_{\mathrm{x}}+1}}, \quad d_{\mathrm{x}_R} = \frac{2(2k_{\mathrm{x}}+1)+1}{2^{j_{\mathrm{x}}+1}} \\ d_{\mathrm{y}_L} &= \frac{2(2k_{\mathrm{y}}+1)-1}{2^{j_{\mathrm{y}}+1}}, \quad d_{\mathrm{y}_R} = \frac{2(2k_{\mathrm{y}}+1)+1}{2^{j_{\mathrm{y}}+1}} \end{aligned} \tag{3.20}$$

The subscripts "L" and "R" denote the left and right child respectively. This method of defining 1-dimensional children follows from the simple association of nodes of a binary tree with dyadic rationals in the unit interval $(0,1]$ as shown in figure (3.18). The root node ($\mathrm{d} = \frac{1}{1}$) corresponds to the scaling function, which captures the low-pass portion of the image. Equation (3.19) holds for children of all nodes other than the root node. Except for the root node, each node has a left and right child and this is expressed in equation (3.20).

The children of a 2-dimensional scale-space position d = (d_x, d_y) are obtained by dilating the corresponding wavelet (i) in x or (ii) in y or (iii) in both x and y. Thus, the children of d can be expressed as

$$\begin{aligned} \text{chil}(d) = \{&(d_{x_L}, d_y),\ (d_{x_R}, d_y),\ (d_x, d_{y_L}),\ (d_x, d_{y_R}),\\ &(d_{x_L}, d_{y_L}),\ (d_{x_L}, d_{y_R}),\ (d_{x_R}, d_{y_L}),\ (d_{x_R}, d_{y_R})\} \end{aligned} \tag{3.21}$$

A graphical illustration of this parent-child relationship is shown in figure (3.19). With this scheme, a scale-space position may have up to 3 parents and up to 8 children. This is easily illustrated with the following example. The node $\breve{d} = (d_{x_R}, d_{y_R})$ has three parents. As is obvious from equation (3.21), one parent is (d_x, d_y) because dilating this in both x and y would result in the child $\breve{d}$. The other parents are the ones from which $\breve{d}$ can be obtained by dilation in only x or y. These are (d_{x_R}, d_y) and (d_x, d_{y_R}) respectively.

We are now ready to define the data structure which we call a web. We call a set D of scale-space positions a "web" if for every scale-space position d in D, at least one parent of d is in D. A web is thus similar to a tree except that a child may have multiple parents.

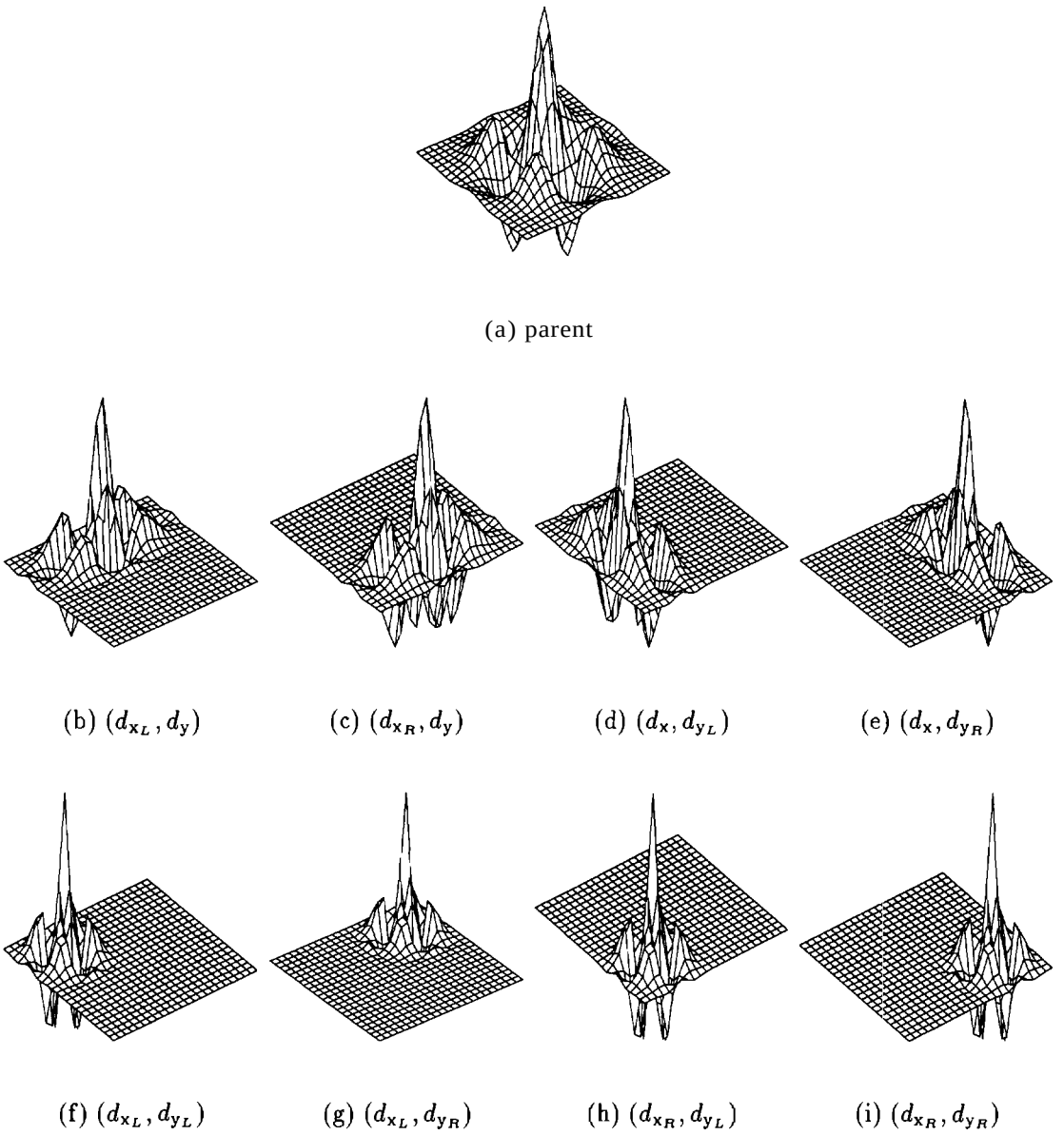
All experimental results in this thesis (with the exception of the comparison in section 3.9) use the TAS algorithm in conjunction with the web. Note that the TAS algorithm can be used with any definition of parent-child relationship. The parent-child relation that we have found most effective is the one defined by the web.

3.9 Web offers more flexibility than a Tree

The TAS algorithm may be used with any parent-child relationship that indexes scale-space positions. The web (defined in section 3.8) is one possible way to define a parent-child relationship. Another way is the quadtree [3] where the children of a node d are defined as

$$\text{chil}_{\text{qtree}}(d) = \{(d_{x_L}, d_{y_L}),\ (d_{x_L}, d_{y_R}),\ (d_{x_R}, d_{y_L}),\ (d_{x_R}, d_{y_R})\} \tag{3.22}$$

It can be verified that the each node has a unique parent.



(a) parent

(b) (d_{x_L}, d_y) (c) (d_{x_R}, d_y) (d) (d_x, d_{y_L}) (e) (d_x, d_{y_R})

(f) (d_{x_L}, d_{y_L}) (g) (d_{x_L}, d_{y_R}) (h) (d_{x_R}, d_{y_L}) (i) (d_{x_R}, d_{y_R})

Fig. 3.19. A 2-dimensional wavelet is product of two 1-D wavelets; as shown in figure (a). Its children are shown in figures (b - i). The children in figures (b, c) are obtained by dilating in x, figures (d, e) by dilating in y and figures (f, g, h, i) by dilating in both x and y.

The drawback of the quadtree, when used with the TAS algorithm, is that it does not offer enough flexibility. Since each node in a quadtree has a unique parent, it is possible that a parent with small magnitude "locks up" a child with large magnitude. In other words, the TAS algorithm cannot get to the child until it has chosen the parent.

In figures (3.20, 3.21), we show the plots that result when the TAS algorithm is used together with the quadtree-based parent-child relationship. The value of 7 is very low ($\gamma_{\text{qtree}} = 0.3$) although the corresponding value for the web-based TAS was much higher ($\gamma_{\text{web}} = 0.9$). As is clear from the figures, certain large coefficients appear very late in the TAS ordering. For example, in figure (3.20), some large coefficients have a TAS-index of roughly 2700. In figure (3.21), the same data is observed as a thick (nearly) horizontal line starting at $y = 2700$. The essential reason why we see this happen is because there is only one parent for these nodes and that parent had small magnitude. So although certain children had large magnitude, they were not chosen early by the TAS algorithm because the parent had small magnitude (and hence was not considered important enough to be chosen).

In contrast, the web assigns three parents for each node. So it is very unlikely that a node with large magnitude has all three parents with small rnagnitude. Most likely, at least one of the parents has large enough magnitude to get chosen early. This flexibility offered by the web makes it a superior data-structure for use with the TAS algorithm. This is clear when we compare figures (3.20, 3.21) to figures (3.4, 3.5).

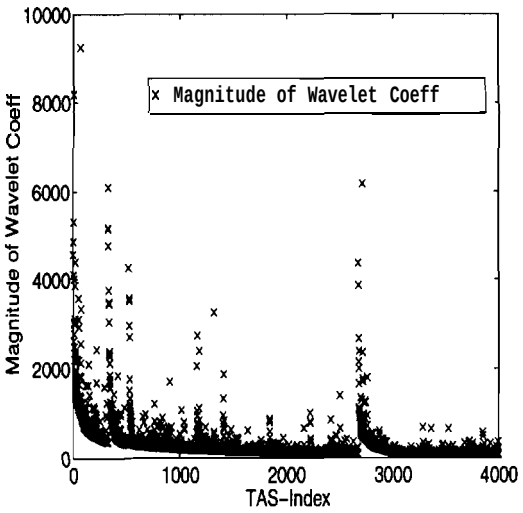


Fig. **3.20.** Plot of coefficients in the order generated by the TAS algorithm when the quadtree is used (instead of a web) for a parent-child relationship.

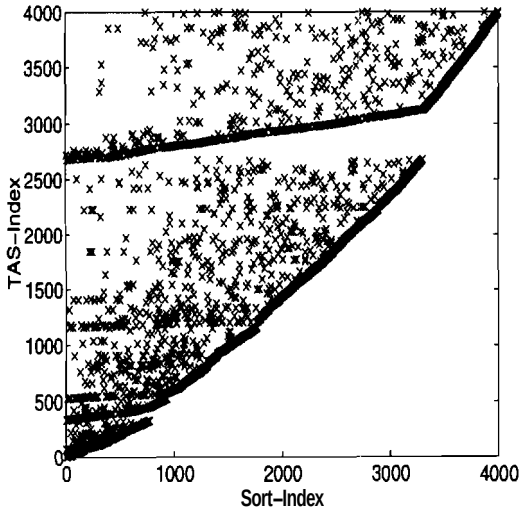


Fig. **3.21.** Plot of TAS-Index versus Sort-Index when the quadtree is used (instead of a web) for a parent-child relationship.

4. A WEB OF WAVELETS FOR IMAGE COMPRESSION

Various properties of wavelet coefficients were outlined in the previous chapter. We now discuss how these properties can be used for image compression.

4.1 Features of the Algorithm

A compressed representation having about 20,000 non-zero coefficients is usually very close to the original in visual quality [3, 4]. As suggested by DeVore et al [4], the number of non-zero coefficients as well as the PSNR are reasonable measures of the quality of the compressed representation. We would therefore like our algorithm to enable us to control both the number of significant coefficients and the PSNR. Further, we would like the algorithm to have low computational requirements by computing only the required coefficients as opposed to all N^2 coefficients for an N x N image.

Let us assume that we have decided to have L_{nz} non-zero coefficients in the represenation. A typical value for L_{nz} may be between 10,000 and 20,000. We will now present an algorithm that has the following important features:

- It is possible to control *both* (i) the number of significant coefficients in the representation and (ii) the PSNR.
- Its performance is comparable to the best available algorithms.
- An upper bound on the number of coefficients to be computed is $\min(8L_{nz}, N^2)$. A typical value for this number is $\min(4L_{nz}, N^2)$. This is in contrast to most wavelet based algorithms where the number of coefficients to be computed is N^2. This is of concern because the computational expense of this task increases dramatically as we move from 512 × 512 to 1920 x 1152 or even larger sized

images.[1] Therefore, computational efficiency may play a significant role in the choice of an algorithm.

- The bitstream of the compressed representation is *embedded.* As defined by Shapiro [3], an embedded representation is one that has all lower rate codes "embedded", in the beginning of the bitstream.

4.2 Overview of Algorithm

We now briefly describe how the algorithm works. As suggested in figure (3.6), we split the web into two parts for the purpose of encoding. The first part consists of coefficients whose sort-index is less than some number L_{init}, and the second part contains the remaining coefficients. In our experiments, we chose L_{init} to be 500 because it is just past the knee of the curve in figure (3.2). Each of the two parts of the web uses a different technique for encoding the positions, signs and magnitudes. In the first part, the magnitudes are encoded to full precision, i.e. rounded to the nearest integer. In the second part, only an approximation of the magnitudes is encoded.

In section 4.3, we discuss how the first part of the web is encoded. Then, in section 4.4, we come to the main topic of this section, which is encoding the second part of the web. Both these encodings utilize a pre-defined scanning scheme which scans all of N^2 scale-space positions for an N x N image. The important property that this scanning scheme is required to satisfy is that each node is scanned before any of its children. Since the nodes are scanned in a pre-defined order, associated with each node we have a unique "scan order". This scanning scheme is discussed at the end of this chapter.

[1] The current MPEG-2 standard supports images of size 1920 x 1152 [54]. Currently, the "PhotoCD" by Kodak can store images of size 6144 x 4096. The "KAF-6300" CCD imaging sensor by Kodak has a resolution of 3088 × 2056 pixels.

4.3 Coding the first part

4.3.1 Positions

Of the L_{nz} nodes, we take the L_{init} that are largest in magnitude and order them in increasing order of sort-index. Let these sort-indices be n;, i $= 1 \cdots L_{\text{init}}$. We have to encode the positions of these nodes.

We start with a matrix of 1's and 0's where the rows contain the binary representation of a number and the leftmost column corresponds to the highest order bit. We fill the ith row of the matrix with the binary representation of the scan order of the node whose sort-index is n;. Then, we make a bitstream from this matrix column by column. That is, we read out the bits of the first column, then move on to the next column and so on.

4.3.2 Magnitudes

Since the positions are encoded in increasing order of sort-index, the corresponding magnitudes will necessary be decreasing, as shown in figure **(3.2).** It is sufficient to encode the difference in magnitudes. We encode the magnitude of the first node rounded to the nearest integer. For each of the remaining nodes, we encode an integer corresponding to the difference between its magnitude and that of the previous node. In order to ensure that each magnitude is decoded with integer precision, the encoder has to perform this encoding from the viewpoint of the decoder. So what we actually encode is the difference between the magnitude of the current node and the estimated magnitude of the previous node as estimated by a decoder.

4.3.3 Signs

The signs are simply encoded as binary numbers in a straightforward manner.

4.4 Coding the latter part of the Web

Now we address the main topic of this section, which is coding the latter part of the web.

4.4.1 Positions and signs

In the encoding scheme for the latter part of the web, we encode the positions and signs simultaneously. We now describe how this is done. We maintain three sets of nodes:

- C: the candidates. These are nodes which are not yet encoded but are the children of some encoded node.
- E: the encoded nodes. These are the nodes that have already been encoded. We start by setting E to be the set of L_{init} nodes already encoded using the scheme of section 4.3.
- $\mathcal{W}_i$, i $= 1, \cdots,$ Q: nodes in the ith quantization level. (Here, Q is the desired number of quantization levels.) The sets $\mathcal{W}_i$ are described below.

Each set W; is to have L; nodes. To define the number L;, we divide the range of the magnitudes of the $L_{\text{nz}} - L_{\text{init}}$ nodes to be encoded into Q equal levels. The number L; is the number of nodes whose magnitudes lie in the ith level. We can now define the sets W;, i $= 1, \cdots,$ Q, as follows. First list the nodes in order of increasing TAS-index. Set i $= 1$. Let the set $\mathcal{W}_i$ contain the first L_i nodes in the list. Remove these nodes from the list. Increment i and repeat for $i = 1, \ldots,$ Q. This definition of the $\mathcal{W}_i$'s gives us Q levels of quantization.

The algorithm is as shown in figure (4.1). Three symbols are used for the encoding process: "+ve" if the candidate is present with a positive sign, "−ve" if the candidate is present with a negative sign and "A" if the candidate is absent.

We now explain the algorithm of figure (4.1). Line 0 initializes the set E of nodes already encoded. Lines $2 \cdots 15$ encode the positions and signs for one level of quantization. Lines 1 and 16 ensure that this is done for each level i, for i $= 1, 2, \cdots,$ Q. In line 3, the node d_j is the jth node scanned using the scanning scheme alluded to earlier and described in section 4.6. Line 4 checks to see if this node is in C $\bigcup$ W;. Recall that a node in C is a candidate and a node in W; is one whose

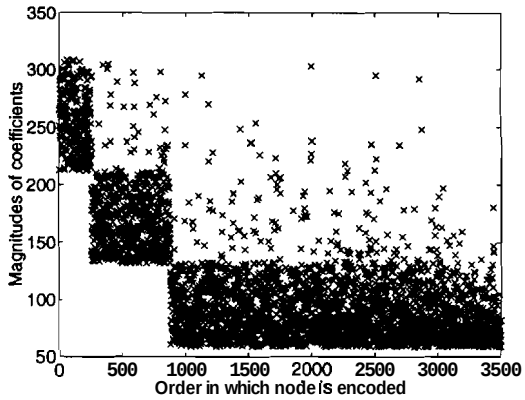
```
Start:  ℰ = {the L_init nodes encoded in section 4.31      0

for i = 1 : Q {                                            1
  for j = 1 : N^2 {                                        2
    Let d_j be the jth node scanned;                       3
    if dj ∈ C ∪ W_i {                                      4
      /*i.e. is a candidate and is to be encoded*/         5
      if coeff has positive sign, then encode "+ve";       6
      if coeff has negative sign, then encode "-ve";       7
      update C :  C = C ∪ ( chil(d_j) \ ℰ );               8
      update ℰ :  ℰ = ℰ ∪ {d_j} ;                          9
      }                                                    10
    else if d_j ∈ C ∪ (W_i)^c {                            11
      /*is a candidate but Absent or has different i*/     12
      encode "A";                                          13
      }                                                    14
    }                                                      15
  }                                                        16
```

Fig. 4.1. Algorithm for coding the main part of the web. Positions and signs are encoded by this part of the algorithm.

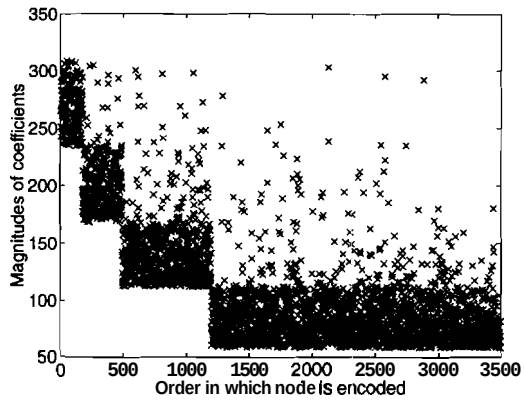
magnitude lies in the ith quantization level. If a node is in both C and $\mathcal{W}_i$ then we encode its presence in the ith quantization level by assigning either a "+ve" or a "−ve" depending on its sign. This is done in lines 6 and 7. Then, in line 8, we update the set C of candidates to include those children of d_j that are not already encoded. In this line, the symbol "\" indicates the "set-difference" operator. So we update C with $\text{chil}(d_j) \bigcap \mathcal{E}^c$, where the superscript "c" denotes set complementation. In line 9, the set $\mathcal{E}$ is updated to include the node d_j. Note that lines $5, \cdots, 10$ happen only when the node is to be encoded for the current value of i. Line 11 considers the possiblity that the node is a candidate but is not to be encoded. This may happen if the node has not been chosen at all by the web or if its magnitude lies in a different quantization level other than i. In this case, we encode the symbol. "A" to denote absence. This completes the encoding algorithm.

It is easy to verify that the above algorithm encodes the position and sign of each of the L_i nodes whose magnitudes lie in the ith quantization level. The outer loop ensures that this is done for each $i = 1, \cdots, Q$. The original TAS-index of the nodes is lost because in line 3, we scan the nodes using the pre-defined scanning scheme. However, some information about TAS-index is retained: We encode the set $\mathcal{W}_i$ in which the node lies. This gives us information about the quantization level in which the node lies.

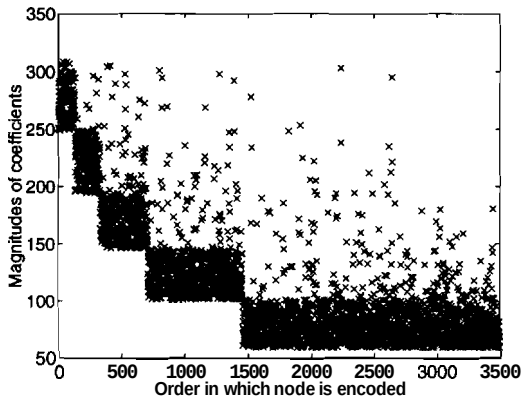
Notice that in the algorithm of figure (4.1), we never consider the possibility of a node not being a candidate, i.e. if it is not in C. The reason is tlnat these nodes can never appear on the web and hence it is unnecessary to scan them. Thus a large number of scale-space positions are eliminated from being scanned and this accounts for a significant proportion of the achieved compression. It is clear that the scale-space positions are scanned in an adaptive manner, adapting to the nature of the image. It is interesting to note that the zerotree algorithm [3] also uses an adaptive scan of the scale-space positions, the adaptation being implemented using a zerotree.



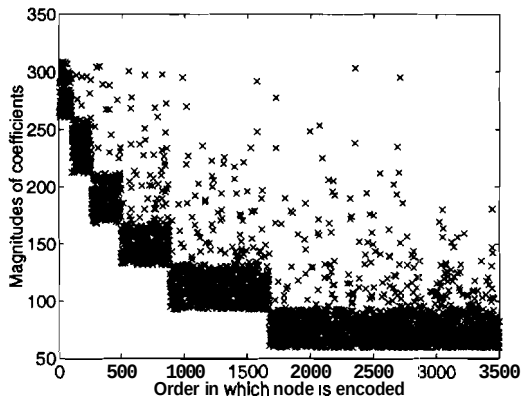
(a) $Q=3$ quantization levels



(b) Q=4 quantization. levels



(c) Q=5 quantization levels



(d) Q=6 quantization levelrr

Fig. 4.2. For each coefficient, assignment of quantization level is done based on the TAS-index rather than the magnitude of the coefficient (i.e. rather than the sort-index). Here, we show various quantization levels.

4.4.2 Magnitudes

We now have to encode the magnitudes of the nodes in the latter part of the web. To do this, we could use the $\frac{1}{x^{\alpha}}$ curve discussed in section 3. However, what we actually use is even simpler: We use a piecewise linear approximation to that curve. So we use a straight line for approximating the data points in each quantization level. We encode the y value (ordinate) of the first and last points of the fitted line-segment for each quantization level. Note that within each level, the first thing that is encoded are these ordinates and only after that is the data for the positions and signs encoded. This ensures that if decoding terminates abruptly, the decoder can reconstruct the image for all positions decoded.

Given a coefficient, the quantization level in which to place that coefficient is usually based on its magnitude [3]. This is exactly equivalent to basing the decision on the sort-index of the coefficient. In the method we described above, this decision is based on the TAS-index rather than the sort-index. This is shown in figure (4.4.1). We use this method to quantize coefficients because encoding becomes more efficient and the resulting error is very small as predicted by figure (3.6).

There is an interesting feature to observe in figure (4.4.1). First, recall that for each coefficient, assignment of quantization levels is done based on the TAS-index rather than the magnitude of the coefficient. (Assignment of quantization level based on magnitude is exactly equivalent to that based on sort-index.) When we use the TAS-index to assign quantization levels, we observe the following feature: About 6% of the coefficients have their magnitudes lying above the limits of their quantization levels whereas none lie below. Therefore, only about 6% of the nodes are affected by the fact that we used a different methodology for assigning quantization levels.

4.5 Lossless encoding

Our method so far has focussed on lossy encoding because for many applications, lossy encoding is sufficient [55, 56, 57, 58]. The reasons why our rnethod is lossy

are (i) the number of quantization levels Q is fixed (by the user), (ii) we use TAS-index rather than magnitude to assign quantization levels, and (iii) all nodes are not encoded.

Our method can be extended for lossless encoding as follows.

1: For each transmitted node, transmit the error in the current approximation of the magnitudes.

2: Transmit more nodes using the method of section 4.4.

Iterate the above steps until all nodes have been transmitted.

Of course, there are many ways in which the above can be implemented. Here, we do not go into the details but merely point out that our method can easily be extended to lossless coding if required.

4.6 Scanning the N^2 nodes

Each of the encoding schemes presented in sections 4.3 and 4.4 relied on a scanning scheme which scans all N^2 nodes of an N x N image. The requirement of this scanning scheme was that a node is scanned before any of its children. The notion of children of a node was defined in equation (3.21).

To formulate the scanning scheme, first recall that in 1 dimension, there are $\max(2^{j-1}, 1)$ scale-space positions at resolution level j. Letting $j = 0, 1, \ldots, J$, there are 2^J positions which have resolution level at most J. We list these level by level, and within each level, from left to right. Now in 2-dimensions, there are 2^{2J} scale-space positions for which both the x and y resolutions are at most J. In figure (4.3), the little dotted circles represent nodes. We have set $J = 3$ and have used the values of resolution j and shift k to represent the nodes (just as in equation (3.12)). We have written the x position along the x axis and the y position along the y axis.

We divide these nodes into four blocks depending on whether the x and y resolutions are less than J or equal to J. So each block has $2^{2(J-1)}$ nodes. We scan these blocks as follows: top left, top right, bottom left, bottom right. Then we decrement J and within each block we follow the same procedure. We continue this until our

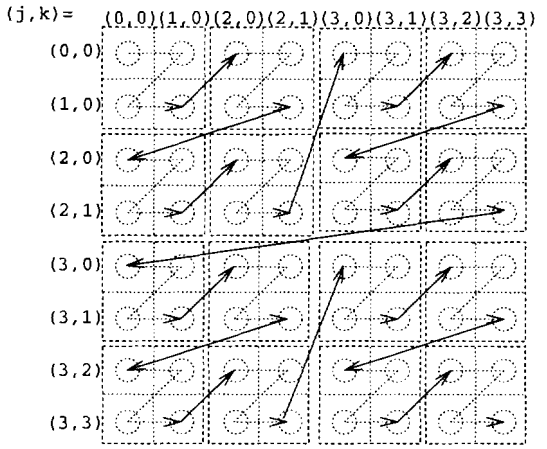


Fig. 4.3. Scanning scheme. Using this scanning scheme, each node is scanned before any of its children.

blocks are of size 1×1 (at which point, J will be 1). The scanning scheme that results is shown in figure (4.3).

Using equation (3.21), it is easy to verify that this scanning scheme scans each node before any of its children. A similar scanning scheme was also proposed in [3]. However, the way in which our compression algorithm uses this scanning scheme is very different from that in [3].

4.7 Experimental Results

The complete encoding and decoding schemes were implemented and tested in order to verify the performance of our algorithm. We present here the results for the "Lenna" and "Couple" images shown in figures (4.4a, 4.5a) respectively. Results are shown for various values of (i) number of significant coefficients and (ii) levels Q of quantization. The number of bits required and the PSNR are also shown. In our experiments, we used the the spline-variant wavelet of [2] because its performance was judged to be "good overall" by Villasenor et al [59]. The entire stream of symbols was arithmetically encoded using a variant of the algorithm in [60].

We can use a given bit budget to either (i) encode the positions of more number of non-zero coefficients or (ii) to encode the magnitudes of existing coefficients with

more precision. If we use too few number of coefficients, then we get blurring of edges as seen in figures (4.4g,h). Roughly speaking, high frequency objects (i.e. edges) get represented as low frequency objects (i.e. blurred edges). On the other hand, if we use too few quantization levels of precision (i.e. small Q) as in figures (4.4d,f,h), then regions which were smooth in the original start looking "wavy" after compression. Roughly speaking, low frequency objects (i.e. smooth regions) get represented as higher frequency objects (i.e. wavy regions). This discussion suggests that a judicious allocation of the bit budget is required when we trade off allocation of bits for encoding the positions and the magnitudes of the coefficients.

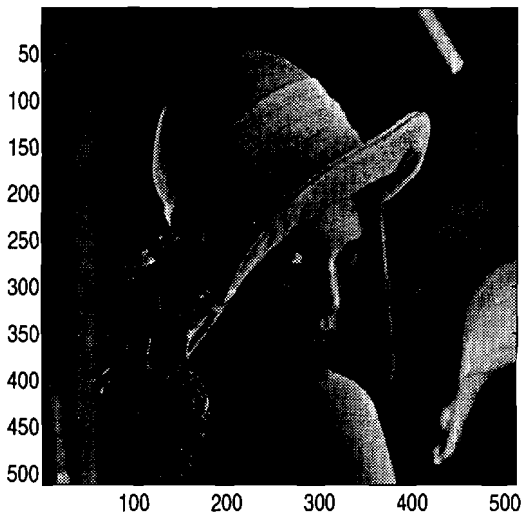
In figure (4.5), we show compression of the "Couple" image. As can be seen from figures (4.5c,d), the features on the girl's face are not visible when 10,000 non-zero coefficients are used but they become visible with 20,000 non-zero coefficients. However, the visual quality does not increase significantly when we go from $Q = 6$ quantization levels to $Q = 10$ quantization levels (see figures (4.5b,c)).

Figure (4.6) shows how the number of bytes required varies for various values of Q and number of coefficients. A comparison to the zerotree algorithm [3] is included. After about 7,000 coefficients, the zerotree algorithm [3] requires more number of bytes to encode the same number of coefficients. The reason probably is that we use only a 3-symbol alphabet whereas they use 4 symbols. In the initial portion of the graph, we have a high overhead associated with coding the first part of the web to full precision. So the zerotree algorithm does better in the initial portion of the graph. Of course, it may be possible to combine the techniques of the zerotree algorithm with ours to reduce this overhead.

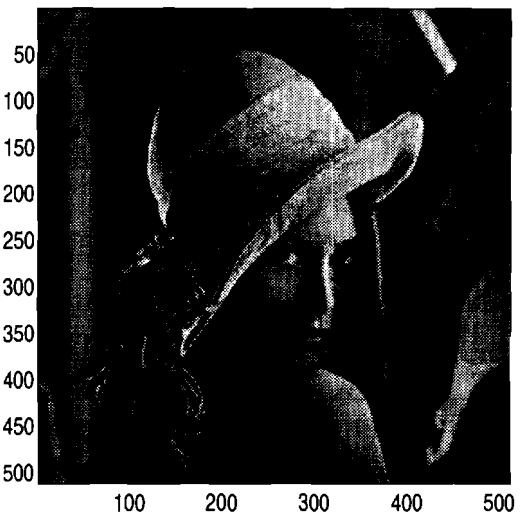
Figure (4.7) shows how the PSNR varies for various values of Q and number of coefficients. An interesting fact is that for low values of Q, the PSNR actually drops as the number of coefficients is increased. This is because we used the TAS-index to assign quantization levels and not the magnitudes (see section 4.4.2 and figure (4.41)) So the error in the estimate of the coefficient may be greater than the coefficient itself.

The main point to be made with figures (4.6, 4.7) is that these figures help us allocate the bit budget. This is because figure (4.6) indicates the possible operating points and figure (4.7) shows the resulting PSNR for these operating points. For example, consider two scenarios: (i) Q=6 and 17,000 coefficients or (ii) Q=7 and 16,000 coefficients. Both require approximately 10,800 bytes to encode. However, this first yields 34.55 dB PSNR whereas the second yields a slightly higher PSNR of 34.61 dB. So (ii) is the better choice if we focus on PSNR, yet (i) is the better choice if we focus on number of coefficients.

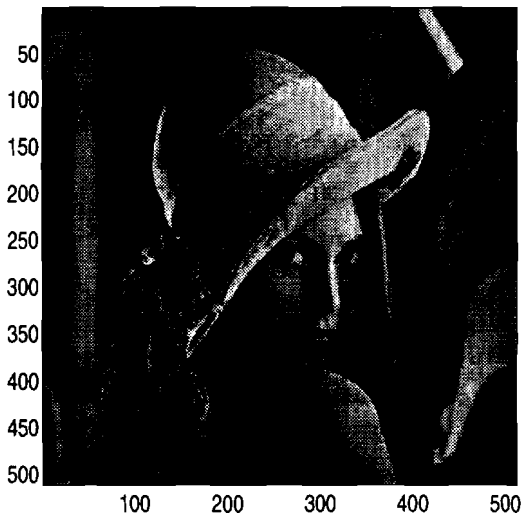
There are numerous results published on the compression of the "Lenna" image [3, 2, 6, 25, 9, 61]. Of these, the embedded zerotree method of Shapiro [3] is of particular interest because it provides close to the best performance for a reasonable computational cost. For 32:1 compression of the "Lenna" image, they quote a PSNR of 33.17 dB, which is almost as good as the 33.95 dB that our algorithm yields. However, the number of significant coefficients they retain is 9774 as compared to 11477 for our method. The difference in the PSNR is minor and can be attributed to various factors, especially the fact that they used a different wavelet. On the other hand, the difference in the number of non-zero coefficients is significant and can possibly be attributed to two factors (i) we use only 3 symbols in our alphabet whereas they require 4, and (ii) we probably use the given bit budget to encode more number of coefficients but with less precision.



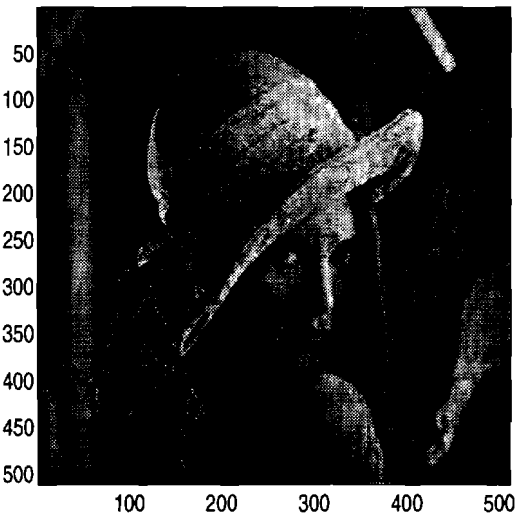
(a) Original image



(b) 11,477 coefficients, Q=6, 8192 bytes, PSNR = 33.95 dB, 32:1 compression

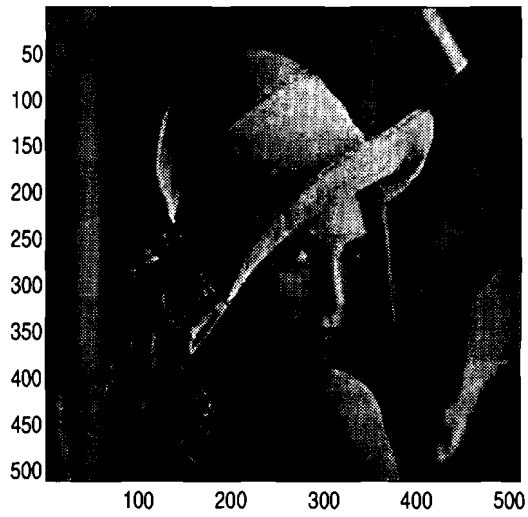


(c) 10,000 coefficients, Q=5, 7172 bytes, PSNR = 33.58 dB, 36:1 compression

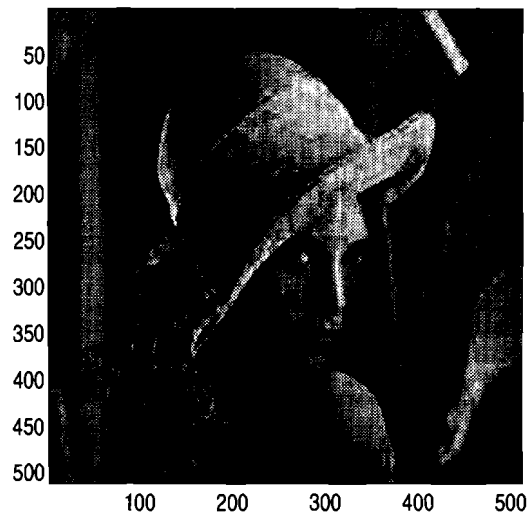


(d) 10,000 coefficients, Q=1, 5878 bytes, PSNR = 30.41 dB, 44:1 compression

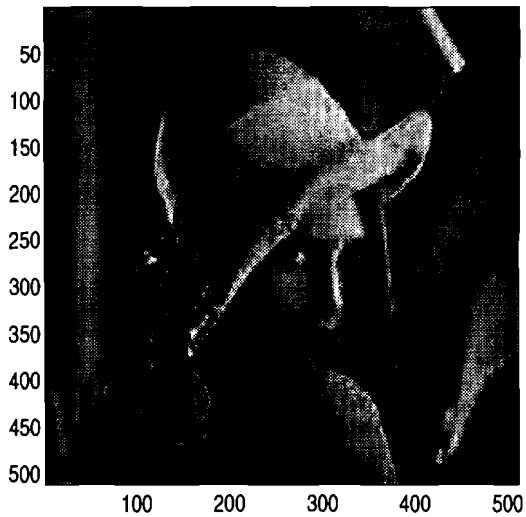
Fig. 4.4. Compression of Lenna: Various values were chosen for (i) number of non-zero coefficients and (ii) number of quantization levels Q. If number of coefficients is too low, edges get blurred. If Q is too low, then smooth regions start to look "wavy". Figure (e) is probably sufficient for browsing purposes.



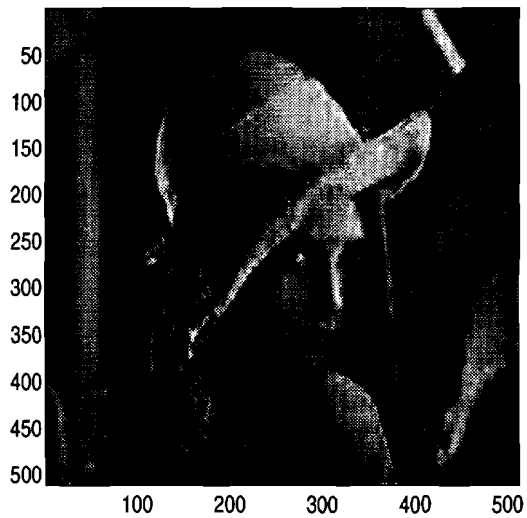
(e) 4,000 coefficients, Q=4, 3657 bytes, PSNR = 31.46 dB, 71:1 compression



(f) 4,000 coefficients, Q=1, 3010 bytes, PSNR = 30.31 dB, 87:1 compression

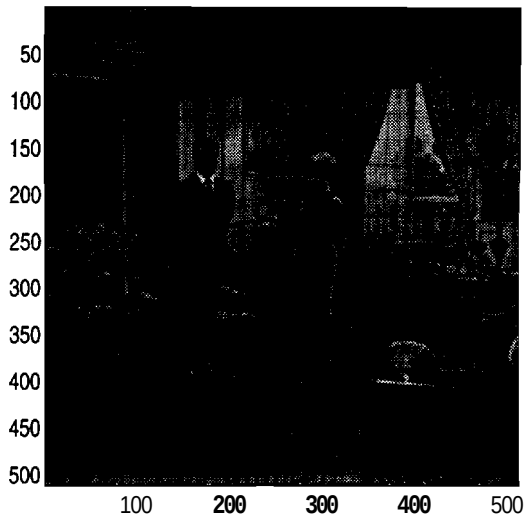


(g) 2,000 coefficients, Q=3, 2257 bytes, PSNR = 29.59 dB, 116:1 compression

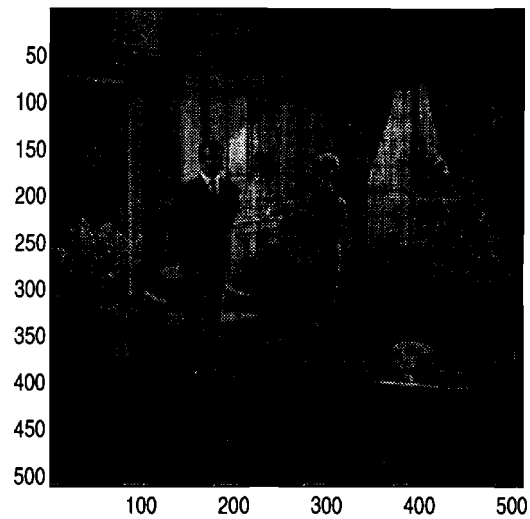


(h) 2,000 coefficients, Q=1, 2032 bytes, PSNR = 29.31 dB, 129:1 compression

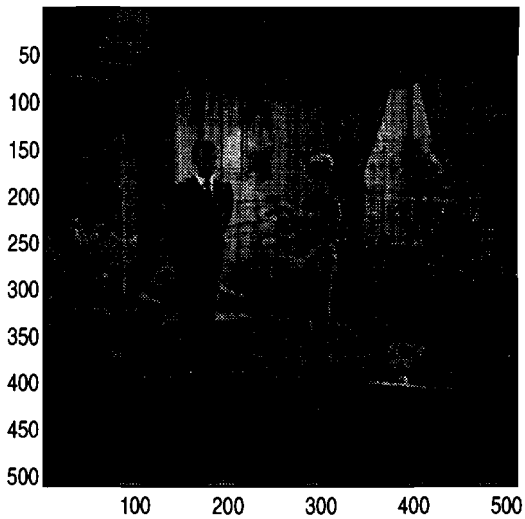
Fig. 4.4. Compression of Lenna: continued.



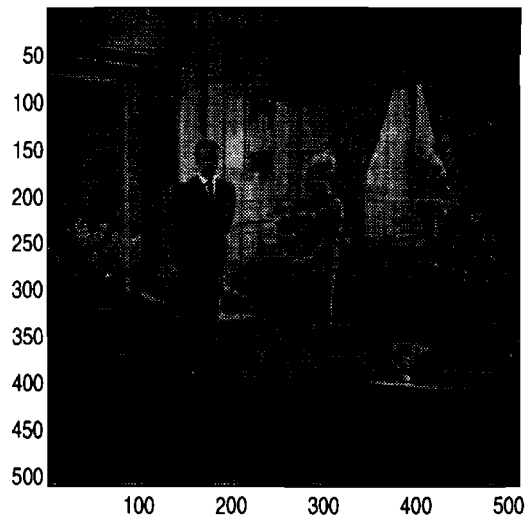
(a) Original image



(b) 20,000 coefficients, Q=10, 14450 bytes, PSNR = 35.15 dB, 18:1 compression



(c) 20,000 coefficients, Q=6, 13017 bytes, PSNR = 34.95 dB, 20:1 compression



(d) 10,000 coefficients, Q=6, 7780 bytes, PSNR = 33.37 dB, 33:1 compression

Fig. 4.5. Compression of the "Couple" image: As can be seen from figures (b,c), not much visual quality is gained by choosing Q higher than 6.

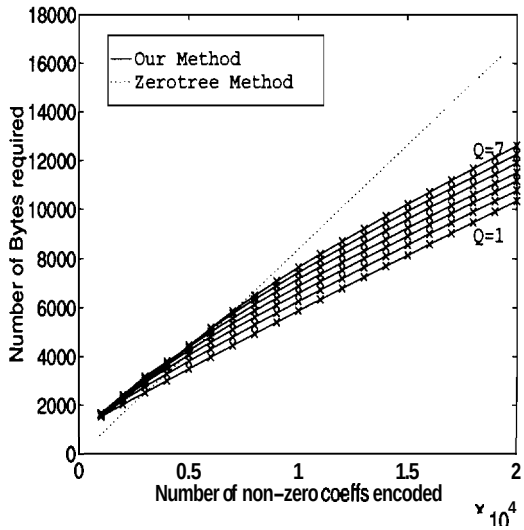


Fig. 4.6. Lenna image: Number of bytes required to encode a given number of coefficients for values of Q from 1 to 7.

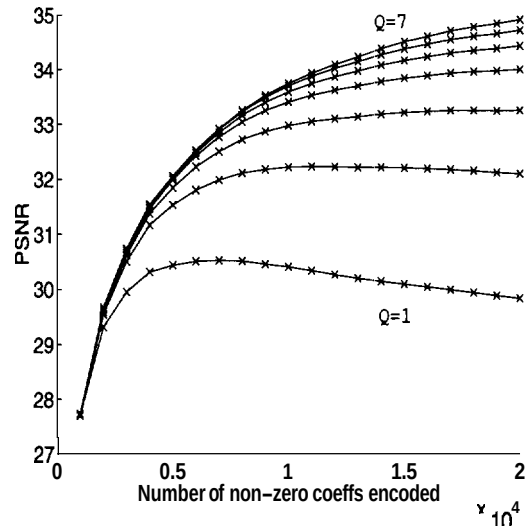


Fig. 4.7. Lenna image: PSNR resulting from encoding; a given number of coefficients for values of Q from 1 to 7.

5. A WAVELET-BASED STOCHASTIC PROCESS FOR IMAGE REPRESENTATION

We now formulate a stochastic process that incorporates the statistical observations presented in chapter 3. This stochastic process provides a theoretical framework in which one may evaluate the performance of an image processing algorithm. In this paper, we investigate some properties of the TAS algorithm when applied to this stochastic process.

Before we describe the construction of the stochastic process, we briefly outline the motivation behind having such a process. The stochastic process we describe here can serve as a stochastic model for images. It

- gives us a theoretical framework in which we can analyse the performance of various image processing algorithms.
- increases our understanding of images.
- gives prior information (prior density) to enable maximum a-priori estimation of an image from noisy data.
- can be used to formulate a joint source-channel image coding scheme.
- can be used to predict the image to subpixel accuracy.

5.1 Construction of the Stochastic Process

The stochastic process we construct is a random function of x, y given by

$$f(x,y) = \sum_{d \in D} c_d \psi_d(x,y) \tag{5.1}$$

where the set D of scale-space positions is random and the coefficients c_d are also random.

We now describe how the random set D is constructed. The construction is iterative, and the $k+1$th scale-space position is chosen randomly from the set chil(D_k), which is the set of children of the previously chosen k positions. This is done as follows. Set $D_1 = \{$ the root node $\}$. Let $D_k = \{d_1, d_2, \ldots, d_k\}$ be the set of the first k randomly chosen scale-space positions. Let M_k be the number of children of D_k, i.e. $M_k = |\text{chil}(D_k)|$. We choose d_{k+1} randomly from chil(D_k) with a uniform conditional distribution:

$$P(d_{k+1} = \breve{d} \;/\; D_k) = \begin{cases} \frac{1}{M_k}, & \breve{d} \in \text{chil}(D_k) \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

Thus, in our stochastic process, the set D of scale-space positions is random and ordered, with a random ordering governed by (5.2).

It now remains to assign a coefficient c_d to each scale-space position d. Using equation (3.1), the magnitude of the mth coefficient c_{d_m} can be written as

$$|c_{d_m}| \;=\; \frac{C}{(\frac{m}{\gamma} - \nu(m) + \triangle)^{\alpha}} \tag{5.3}$$

where C, a and $\triangle$ are as defined in equation (3.1) and $\nu(m)$ is an appropriately chosen random variable. Notice that $\nu(m)$ must satisfy the following properties:

1. I_{sort} (m) $= \frac{m}{\gamma} - \nu(m)$ is a positive integer.
2. $\nu(m)$ is non-negative. (This follows from equation (A.1) in the appendix.)
3. $\nu(\cdot)$ defines a one to one map from the positive integers $\mathbf{Z}^+$ to $\mathbf{Z}^+$ (because it maps TAS-indices m to sort-indices I_{sort} (m)).

It now remains to assign the signs of the coefficients. To do this, we employ the prediction method suggested in section 4:

$$\begin{aligned} &\text{if} \quad s_k^+ < s_k^-, \quad \text{then choose } + \text{ ve sign} \\ &\text{if} \quad s_k^+ \geq s_k^-, \quad \text{then choose } - \text{ ve sign} \end{aligned} \tag{5.4}$$

where s^+ and s^- are as defined in equation (3.17). This completes the definition of the stochastic process.

5.2 Properties of the Stochastic Process

The theorems in this section will assume that the stochastic process is constructed using orthonormal wavelets. Some of the results can be extended to biorthogonal wavelets as well, but in the interest of clarity, we only discuss the orthonormal case.

Theorem 3 : There exists a stochastic process as described by equations (5.2, 5.3, 5.4).

Proof: The construction of the stochastic process is straightforward except for the definition of the random variable $\nu(m)$ in equation (5.3). A detailed proof that $\nu(m)$ can be appropriately constructed is given in the appendix. □

Theorem 4 : The stochastic process has finite energy if and only if $\alpha > \frac{1}{2}$.

Proof: since $\mathrm{f} = \sum_d c_d \psi_d$ and the ψ_d's are orthonormal, the energy of f can be expressed as

$$\|f\|^2 = \sum_d |c_d|^2 = \sum_{m=1}^{\infty} \left(\frac{C}{(\frac{m}{\gamma} - \nu(m) + \triangle)^\alpha} \right)^2 = \sum_{n=1}^{\infty} \left(\frac{C}{(n + \triangle)^\alpha} \right)^2 \quad (5.5)$$

where the last equality follows from the fact that ν of equation (5.3) defines a one to one map from $\mathbf{Z}^+$ to $\mathbf{Z}^+$. It is well known [62] that the last line of equation (5.5) converges if and only if $\alpha > \frac{1}{2}$. □

We will now prove a theorem which guarantees that the amount of detail provided by the stochastic process is infinite. So the discernability of detail in this process is always limited by physical limitations of the observation procedure rather than the stochastic process itself. This is a desirable property for a stochastic model of an image.

Theorem 5 : Let $(\mathrm{z}, \mathrm{y}) \in (0,1]$ x $(0,1]$ and let $\epsilon > 0$ be an arbitrarily small positive number. Then with probability 1 (w.p. 1), the stochastic process adds details infinitely often (i.o.) inside any region $r_{x,y} = (x-\epsilon, x+\epsilon) \times (\mathrm{y}-\epsilon, \mathrm{y}+\epsilon)$ of $(0,1] \times (0,1]$.

Proof: The proof is in the appendix. □

The practical implication of this theorem is that we can use the stochastic process to predict the image to subpixel accuracy. This can be done as follows. First run the algorithm for about 4000 nodes and estimate α. Then continue running the algorithm, and for the nodes of subpixel accuracy, assign coefficients randomly using equation (5.3).

5.3 Limitations of the Process

The properties that we have outlined so far appear to be *necessary* conditions that the wavelet transform should obey for the given data to visually resemble a natural image. In other words, if the wavelet transform of a given 2-dimensional data set does not obey the various properties of sections 2, 3 and 4, then we may conclude that the given data is unlikely to be a natural image.

However, the conditions outlined in chapter 3 do not appear to be *sufficient* conditions. The reason is the following: Instantiations of the stochastic process (shown in figure 5.1) do not seem to resemble a natural image although all the properties of sections 2, 3 and 4 have been incorporated in the definition of the process. In fact, some instantiations of the process more closely resembled certain fractal images such as the one in [63, plate 15]. While this fact is interesting in itself, it still remains to formulate a stochastic model that incorporates all aspects of natural images. In other words, it appears that there are certain statistical properties of the wavelet coefficients of natural images that are crucial, but as yet unknown.

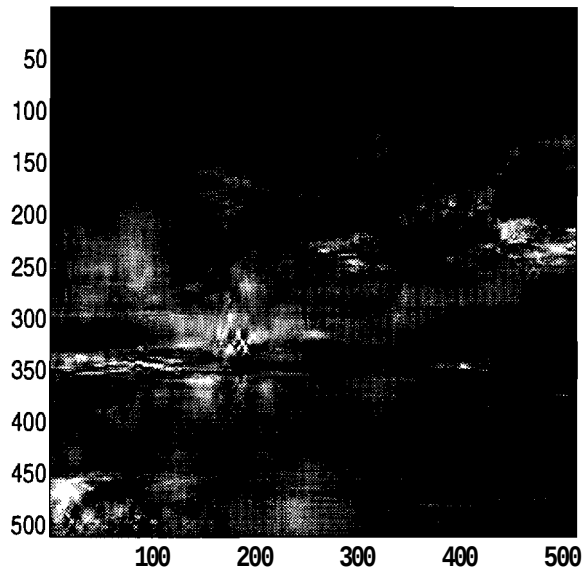


Fig. 5.1. An instantiation of the stochastic process.

6. CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

We performed a detailed study of the statistical structure present in the wavelet decomposition of natural images. Using the results of this study, we proposed an image compression algorithm as well as a wavelet-based stochastic process to model an image.

We showed that self-similarity may exist in the wavelet domain. even if no such property is observable in the pixel domain. The reason for this is that the wavelet coefficients, when ordered by magnitude, lie on a curve of the form $\frac{1}{x^\alpha}$, and this curve has a scale-invariant property.

We performed a detailed study of a top-down greedy search algorithm for selecting wavelet coefficients. We found that this method of choosing coefficients is very similar to choosing them in decreasing order of magnitude. We founcl a bound on the error between the two approaches. We showed that the top-down greedy algorithm automatically zooms in on the edges present in the image. We also addressed the issue of predicting the signs of wavelet coefficients.

These observations were used to formulate an image compression algorithm. This scheme for image compression offers the user the ability to control the number of non-zero coefficients as well as the PSNR. It has low computational requirement and good performance.

Based on our empirical observations, we proposed a stochastic process to model an image. We studied various theoretical properties of the process. Finally, we examined some limitations of the proposed stochastic process.

6.2 Future Work

There are numerous details in both the empirical observations as well as the theoretical formulation which can be refined. For example, the prediction of signs has not yet been linked to the prediction of scale-space positions and magnitudes. Also, using an autoregressive model, it may be possible to perform a better prediction of which scale-space positions correspond to large coefficients.

An important stochastic property observed in **[3]** is that if a wavelet coefficient is small, then it is very likely that all its children are small. We verified experimentally that roughly 80% of the time, the children have smaller magnitude than the parents. However, we have not been able to incorporate this property into our proposed stochastic process. Further, we have not shown theoretically that this property directly follows from the definition of our stochastic process. This can be studied in the future.

Another important extension of our algorithm will be to use it for video compression. The essential method we will study is to be able to predict the web corresponding to the present frame from that of the previous frame. Other extensions, such as joint source-channel coding can also be explored.

LIST OF REFERENCES

[1] K. R. Rao and P. P. Yip, "Discrete Cosine Transform: *algorithm,* advantages, applications ", Academic Press, New York, 1990.

[2] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, ""Image Coding using Wavelet Transform"", IEEE Transactions on Image Processing, vol. 1, pp. 205 – 220, April 1992.

[3] J. Shapiro, ""Embedded Image Coding Using Zerotrees of Wavelet Coefficients"", IEEE Transactions on Signal Processing, vol. 41, pp. 3445 – 3462, December 1993.

[4] R. DeVore, B. Jawerth, and B. Lucier, ""Image Compression through Wavelet Transform Coding"", IEEE Transactions on Information Theory, vol. 38, pp. 719 – 746, March 1992.

[5] I. Katsavounidis and C. J. Kuo, ""Image compression with embedded wavelet coding via vector quantization"", in SPIE Conference on *Mathematical* Imaging, pp. 333 – 344, San Deigo, California, 1995.

[6] P. Sriram and M. W. Marcellin, ""Image coding using wavelet transforms and entropy-constrained trellis-coded quantization"", IEEE *Transactions* on Image Processing, vol. 4, pp. 725 – 733, June 1995.

[7] Z. Xiong, K. Ramachandran, and M. Orchard, ""Joint optimization of scalar and tree-structured quantization for wavelet image decomposition"", in Proceedings of the *27th* Asilomar Conf. Sig. Sys. Comp., Pacific Grove, CA, Nov 1993.

[8] T. Strutz and E. Muller, ""Image data compression with pdf-adaptive reconstruction of wavelet coefficients"", in SPIE Conference on Mathematical Imaging, pp. 747 – 758, San Deigo, California, July 1995.

[9] K. A. Birney and T. R. Fischer, ""On the Modeling of DCT and Subband Image Data for Compression"", IEEE Transactions on Image Processing, vol. 4, pp. 186 – 193, February 1995.

[10] M. P. Barnsley and L. P. Hurd, "Fractal Image Compression ", A. K. Peters, Massachusetts, 1993.

[11] Y Fisher, "Fractal Image Compression", Springer Verlag, New York, 1995.

[12] A. Jaquin, ""Image coding based on a fractal theory of iterated contractive image transformations"", IEEE Transactions on Image *Processing*, vol. 1, pp. 18 – 30, January 1992.

[13] R. C. Reininger and J. D. Gibson, ""Distributions of the two-dimensional DCT coefficients for images"", IEEE Transactions on *Communications*, vol. COM-31, pp. 835 – 839, June 1983.

[14] K. Abend, T. J. Hartley, and L. N. Kanal, ""Classification of Binary Random Patterns"", IEEE Transactions on Information Theory, vol. 11, pp. 538 – 544, October 1965.

[15] R. Chellappa and R. L. Kashyap, ""Image Restoration using Spatial Interaction Models"", IEEE Transactions on *Accoustics,* Speech and Signal Processing, vol. 30, pp. 461 – 472, June 1982.

[16] R. L. Kashyap and K. Eom, ""Robust Image Models and Their Applications "", in P. W. Hawkes, editor, Advances in Electronics and Electron Physics, Vol *70*, pp. 80 – 157. Academic Press, 1988.

[17] B. S. Manjunath and R. Chellappa, ""Unsupervised texture segmentation using Markov random field models"", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 13, pp. 478 – 482, 1991.

[18] M. R. Luttigen, W. C. Karl, A. S. Willsky, and R. R. Tenney, ""Multiscale representations of Markov random fields"", IEEE Transactions on Signal Processing, vol. 41, pp. 3377 – 3396, December 1993.

[19] C. Bouman and B. Liu, ""Multiple Resolution Segmentation of Textured Images"", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 13, pp. 99 – 113, February 1991.

[20] C. Bouman and M. Shapiro, ""A multiscale random field model for Bayesian image segmentation"", IEEE Transactions on Image *Processing*, vol. 3, 1994.

[21] J. Besag, ""On the Statistical Analysis of Dirty Pictures "", Journal of the Royal Statistical Society, vol. 48, pp. 259 – 302, 1986.

[22] S. Geman and D. Geman, ""Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images"", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 6, pp. 721 – 741, November 1984.

[23] D. Tretter, C. A. Bouman, K. W. Khawaja, and A. A. Maciejewski, ""A Multiscale Stochastic Image Model for Automated Inspection"", *IEEE* Transactions on Image Processing, vol. 3, 1994.

[24] X. Ran and N. Farvardin, ""A Perceptually Motivated Three-Component Image Model-Part I "", IEEE Transactions on Image Processing, vol. 4, pp. 401 – 415, April 1995.

[25] X. Ran and N. Farvardin, ""A Perceptually Motivated Three-Component Image Model-Part II "", IEEE Transactions on Image Processing, vol. 4, pp. 430 – 447, April 1995.

[26] W. Hwang and H. Derin, ""Multiresolution Multiresource Progressive Image Transmission"", IEEE Transactions on Image Processing, vol. 4, pp. 1128 – 1140, August 1995.

[27] A. K. Jain, "Fundamentals of Digital Image Processing", Prentice Hall, New Jersey, 1989.

[28] P. N. Topiwala and R. D. Forkert, ""Region of Interest Compression Using Pyramidal Coding Schemes "", in SPIE Conference on *Mathematical* Imaging, pp. 759 – 765, San Deigo, California, July 1995.

[29] I. Daubechies, "Ten Lectures on Wavelets", SIAM, Philadelphia, 1992.

[30] C. K. Chui, "An Introduction to Wavelets", Academic Press, San Diego, California, 1992.

[31] M. Vetterli and J. Kovacevic, "Wavelets and *Subband* Coding", Prentice Hall, New Jersey, 1995.

[32] G. Strang and T. Nguyen, "Wavelets and Filter Banks", Wellesley Cambridge Press, Wellesley, MA, 1996.

[33] G. Walter, "Wavelets and Other Orthogonal Systems With Applications", CRC Press, Boca Raton, Florida, 1994.

[34] Y. Meyer, "Wavelets: Algorithms and Applications ", Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1993.

[35] B. Jawerth and W. Sweldens, ""An Overview of Wavelet Based Multiresolution Analyses"", SIAM Review, vol. 26, pp. 377 – 412, September 1994.

[36] R. S. Strichartz, ""How to Make Wavelets"", The American Mathematical Monthly, vol. 100, pp. 539 – 556, June-July 1993.

[37] O. Rioul and M. Vitterli, ""Wavelets and Signal Processing"", IEEE Signal Processing Magazine, vol. 8, pp. 14 – 38, October 1991.

[38] S. G. Mallat, ""Multiresolution Approximations and Wavelet Orthonormal Bases of $\mathcal{L}^2(\mathbf{R})$"", Transactions of the American Mathematical Society, vol. 315, pp. 69 – 87, 1989.

[39] M. Basseville, A. Benveniste, K. C. Chou, S. A. Golden, R. Niloukhah, and A. S. Willisky, ""Modeling and Estimation of Multiresolution Stochastic Processes"", IEEE Transactions on Information Theory, vol. 38, pp. 766 – 784, March 1992.

[40] K. Ramachandran and M. Vitterli, ""Best Wavelet Packets in a. Rate-Distortion Sense"", IEEE Transactions on Image Processing, vol. 2, pp. 160 – 175, April 1993.

[41] C. TaswellI, ""Top-Down and Bottom-up Tree search Algorithms for Selecting Bases in Wavelet Packet Transforms"", in A. Antoniadis and G. Oppenheim, editors, Wavelets and Statistics, pp. 345 – 359. Springer Verlag, 1995.

[42] M. Peruggia, "Discrete Iterated Function Systems ", A. K. Peters, Massachusetts, 1993.

[43] G. W. Wornell, ""Synthesis, analysis, and processing of fractal signals "", Technical Report RLE Tech. Rep. 566, MIT, 1991.

[44] G. W. Wornell, ""A Karhunen-Loeve-like expansion for l/f processes via wavelets"", IEEE Transactions on Information Theory, vol. 36, pp. 859 – 861, July 1990.

[45] P. Abry, P Goncalves, and P. Flandrin, ""Wavelets, spectrum analysis and l/f processes "", in A. Antoniadis and G. Oppenheim, editors, Wavelets and Statistics, pp. 14 – 42. Springer Verlag, 1995.

[46] Y. Fisher and S. Menlove, ""Fractal Encoding with HV Partitions "", in Y. Fisher, editor, Fractal image compression, pp. 119 – 136. Springer Verlag, 1994.

[47] G. E. Oien and S. Lepsoy, ""A Class of Fractal Image Coders with Fast Decoder Convergence "", in Y. Fisher, editor, Fractal image *compression*, pp. 153 – 176. Springer Verlag, 1994.

[48] F. Dudbridge, ""Least-Squares Block Coding by Fractal Functions "", in Y. Fisher, editor, Fractal image compression, pp. 229 – 241. Springer Verlag, 1994.

[49] G. Davis, ""Adaptive self-quantization of wavelet subtrees: .A wavelet-based theory of fractal image compression"", in SPIE Conference on Mathematical Imaging, pp. 294 – 306, San Deigo, California, July 1995.

[50] C. de Boor, "A Practical Guide to Splines", Springer Verlag, New York, 1978.

[51] U. Montanari, ""On the optimal detection of curves in noisy pictures "", Communications of the *ACM*, vol. 14, pp. 335 – 345, 1971.

[52] D. Maar and E. C. Hildreth, ""Theory of Edge Detection"", Proceedings of the Royal Society of London, vol. B270, pp. 187 – 217, 1980.

[53] S. Moni and R. L. Kashyap, ""Image Compression using a Web of Wavelets "", submitted to IEEE Transactions on Image Processing, vol. , January 1996.

[54] R. Aalmoes and P. Bosch, ""Overview of Still-picture and Video Compression Standards"", Technical Report Pegasus-95-03, University of Cambridge, 1995.

[55] P. H. Ang, P A. Ruetz, and D. Auld, ""Video Compression makes Big Gains "", in IEEE Spectrum, pp. 16 – 19. IEEE, 1991.

[56] M. Rabbani and P. W. Jones, "Digital Image Compression *Techniques"*, Tutorial Texts in Optical Engineering, SPIE, Bellingham, WA, 1991.

[57] G. Wallace, ""The JPEG Still Picture Compression Standard"", Communications of the ACM, vol. 34, April 1991.

[58] R. K. Jurgen, ""Digital Video "", in IEEE Spectrum, pp. 24 – 30. IEEE, 1992.

[59] J. D. Villasenor, B. Belzer, and J. Liao, ""Wavelet Filter Evaluation for Image Compression "", IEEE Trans. on Image Proc., vol. 4, August 1995.

[60] I. H. Witten, R. M. Neal, and J. G. Cleary, ""Arithmetic Coding for Data Compression"", Communications of the ACM, vol. 30, pp. 520 – 540, June 1987.

[61] 0. Egger and W. Li, ""Subband Coding of Images using Asymmetric Filter Banks"", IEEE Transactions on Image Processing, vol. 4, pp. 478 – 485, April 1995.

[62] W. Rudin, "Principles of Mathematical Analysis ", McGraw-Hill Book Company, New York, 1976.

[63] H Peitgen and D Saupe, "The Science of Fractal Images ", Springer Verlag, New York, 1988.

[64] S. Moni and R. L. Kashyap, ""Image Representation and Compression using Multisplines and Adaptive Scale-Space Atom Selection"", Technical Report TR-ECE-95-18, Purdue University, 1995.

[65] P. K. Billingsley, "Probability and Measure", John Wiley and Sons, New York, 1986.

APPENDIX A: PROOFS OF THEOREMS

To prove theorem 1, we first prove a lemma.

Lemma 1 : Let I_{sort} (m) to be the sort-index of the wavelet whose TAS-index is m. Then for the AR1 process of equation (3.7), we have $E\{\gamma\} \leq \sum_{n=1}^{N} P(I_{\text{sort}}\ (1) = n) \cdot \frac{1}{n}$.

Proof: Equation (3.3) can be rewritten as

$$\gamma \ \leq \ \frac{m}{I_{\text{sort}}\ (m)} \tag{A.1}$$

For convenience, we define $\gamma_m = \frac{m}{I_{\text{sort}}\ (m)}$. We can therefore write

$$\gamma \ = \ \min_m \gamma_m$$

Let Ω be the sample space of all possible outcomes of the γ_m's and let $w \in \Omega$ be an elementary event. For example, w may represent the event $\{\gamma_1 = \frac{1}{3}, \gamma_2 = \frac{2}{1}, \gamma_3 = \frac{3}{8}, \gamma_4 = \frac{4}{5}, \ldots\}$. Let A be the event defined as

$$A = \{\omega : \gamma_1(\omega) \leq \gamma_i(\omega) \ \ \forall i \geq 2\}$$

In the above equation, we have explicitly denoted the fact the γ_i's (and therefore y) can be viewed as a function of w. Further, we can write

$$\begin{aligned} \gamma(\omega) \ &= \ \gamma_1(\omega) \quad \text{if } \omega \in A \\ \gamma(\omega) \ &\leq \ \gamma_1(\omega) \quad \text{if } \omega \notin A \end{aligned}$$

To derive an upper bound on $E\{\gamma\}$, we write

$$\begin{aligned} E\{\gamma\} \ &= \ E\{\min_m \gamma_m\} \\ &= \ \sum_{\omega \in A} P(\omega)\gamma_1(\omega) + \sum_{\omega \notin A} P(\omega)\gamma(\omega) \end{aligned}$$

$$\begin{aligned} &\leq \sum_{\omega \in A} P(\omega)\gamma_1(\omega) + \sum_{\omega \notin A} P(\omega)\gamma_1(\omega) \\ &= \sum_{\omega \in \Omega} P(\omega)\gamma_1(\omega) \\ &= E\{\gamma_1\} = \sum_{n=1}^{N} P(I_{\text{sort}}(1) = n) \cdot \frac{1}{n} \end{aligned} \tag{A.2}$$

This proves the lemma. □

Proof of Theorem 1

Let Γ_n denote the region in N dimensional space which corresponds to the event that c_1 is the nth largest in magnitude amongst all components c_i of $\mathbf{c}$. Letting $\mathbf{D} = \sigma^2\mathbf{H}\mathbf{A}\mathbf{A}^T\mathbf{H}^T$, we have

$$\begin{aligned} P(I_{\text{sort}}(1) = n) &= P(c_1 \text{ is } n\text{th largest}) \\ &= \int_{\Gamma_n} \frac{1}{(\sqrt{2\pi})^N |\mathbf{D}|^{\frac{1}{2}}} e^{-\frac{1}{2}\mathbf{c}^T\mathbf{D}^{-1}\mathbf{c}}\, d\mathbf{c} \\ &\leq \sqrt{\lambda_{\max}} \int_{\Gamma_n} \frac{1}{(\sqrt{2\pi})^N} \frac{1}{\sigma\sqrt{\lambda_{\max}}} e^{-\frac{\mathbf{c}^T\mathbf{c}}{2\sigma^2\lambda_{\max}}}\, d\mathbf{c} \\ &= \frac{\sqrt{\lambda_{\max}}}{N} \end{aligned} \tag{A.3}$$

The last equality follows from the symmetry that results when $a = 0$ because then the c_i's are independent and identically distributed and each c_i has equal probability of being in the nth spot. Note that since det($\mathbf{A}$)=det($\mathbf{H}$)=1, we have det($\mathbf{D}$) = σ^2. The proof of the theorem is now immediate from equation (A.3) and lemma 1.

Lemma 2 : There exists a non-trivial random sequence of positive integers $\{v_i\}_{i\geq 1}$ which satisfies $v_m \leq \frac{m}{\gamma}$, where $0 < \gamma \leq 1$.

Proof: Since $0 < \gamma \leq 1$, we may trivially construct the required sequence by setting $v_i = i$. In order to construct a map that more closely resembles the relationship between sort-index and TAS-index, we will start with the trivial map and then iteratively modify it. After the mth iteration, we may set $I_{\text{sort}}(i) = v_i, i = 1, 2, \ldots, m$.

At each iteration,the sequence $\{v_i\}$ will define a one to one map from the positive integers $\mathbf{Z}^+$ back to $\mathbf{Z}^+$. So if we have $v_i = k$, then we will say that $v_k^{-1} = i$.

At the mth iteration, we will define a set of positive integers

$$B_m = \{k \in \mathbf{Z}^+ : k \leq \frac{m}{\gamma} \text{and } k \neq v_i \text{for any } i = 1, 2, \ldots, m-1\} \qquad (A.4)$$

Since each element of B_m must be a positive integer between 1 and $\frac{m}{\gamma}$, B_m is a finite set.

The iterative construction of the one to one map between sort-index and TAS-index proceeds as follows:

For m $\geq$ 1

1. Define B_m as in equation (A.4).
2. Randomly pick an element l from B_m.
3. Set $l_{\text{tmp}} = v_m$, $m_{\text{tmp}} = v_l^{-1}$, $v_m = l$, $v_{m_{\text{tmp}}} = l_{\text{tmp}}$.

In the mth iteration of the above procedure, at most two values of the sequence $\{v_i\}$ are changed: These are v_m and $v_{m_{\text{tmp}}}$. The last step within the mth iteration is to swap the values of v_m and $v_{m_{\text{tmp}}}$. Therefore the map $\{v_i\}$ remains a one to one map after each iteration.

The above procedure generates a random one to one map which satisfies $v_m \leq \frac{m}{\gamma}$. Note that with the above procedure, equality is possible even if $\gamma \leq 1$.□

Proof of Theorem 2

Assume there are no edges passing through p. Since the number of edges is finite, we can find an '6-neighborhood of p (call it $B_{p,\epsilon}$) in which there are no edges. So $B_{p,\epsilon} \subset \mathcal{I}_l$ for some $1 \in \{1, 2, \ldots, L\}$. Since the analysing wavelet has regularity. M, we have

$$c_d = \int \psi_d(x, y)\, g(x, y)\, dxdy = 0$$

whenever ψ_d is supported within $B_{p,\epsilon}$. So there are at most a finite number of nodes d $\in B_{p,\epsilon}$ with $|c_d| > 0$ which can be chosen by the TAS algorithm. This concludes the proof. □

We now address the proof of theorem 5. As always, we assume that the image is supported over the unit square (0, 1] x (0, 1]. We let r $\subset$ (0, 1] x (0, 1] denote any region inside the unit square.

Definition Let f_k be the kth approximation of an image f. We say that *detail is added* in a region $r \subset (0,1]\ x\ (0,1]$ if

$$f_{k+1} = f_k + c_d \psi_d$$

where $c_d \neq 0$ and there exists an open ball in $r \bigcap \text{support}(\psi_d)$.

Theorem 5 : Let (x, y) $\in$ (0, 1] x (0, 1] and let $\epsilon > 0$ be an arbitrarily small positive number. Then with probability 1 (w.p. 1), the Ordered Web process adds details infinitely often (i.o.) inside any region $r_{x,y} = (x-\epsilon, x+\epsilon)\ x\ (y-\epsilon, y+\epsilon)$ of (0, 1] x (0, 1].

Proof: The following are the key points used in the proof.

i. For any web D, at least one child d of D has the property that

$$\exists \text{ an open ball in } r_{x,y} \bigcap \text{support}(\psi_d)$$

This simple fact is proved in [64].

ii. $\sum_{k=1}^{\infty} \frac{1}{k} = \infty$

iii. The Borel-Cantelli Lemma [65].

We now start the proof. At the kth stage, the chosen node is d_k and the resulting web is D_k. The set of children of the web D_k is chil(D_k), which we write as

$$\text{chil}(D_k) \ = \ \{b_1^{(k)}, b_2^{(k)}, \cdots, b_{|\text{chil}(D_k)|}^{(k)}\} \tag{A.5}$$

where $|\text{chil}(D_k)|$ is the number of nodes in chil(D_k).

We now form a sequence $\widetilde{\text{chil}}(D_k)$ by appending a copy of chil(D_k) to itself:

$$\widetilde{\text{chil}}(D_k) \ = \ \tilde{b}_1^{(k)}, \tilde{b}_2^{(k)}, \tilde{b}_3^{(k)}, \cdots, \tilde{b}_{2|\text{chil}(D_k)|}^{(k)} \tag{A.6}$$

$$\text{where } \tilde{b}_i^{(k)} \ = \ \begin{cases} b_i^{(k)} & \text{if } 1 \leq i \leq |\text{chil}(D_k)| \\ b_{i-|\text{chil}(D_k)|}^{(k)} & \text{if } |\text{chil}(D_k)| + 1 \leq i \leq 2|\text{chil}(D_k)| \end{cases} \tag{A.7}$$

We now assign probabilities to each of the $\tilde{b}_i^{(k)}$'s as follows:

$$P(\tilde{b}_i^{(k)}) \ = \ \begin{cases} \frac{1}{8k} & \text{if } 1 \leq i \leq |\text{chil}(D_k)| \\ \frac{1}{|\text{chil}(D_k)|} - \frac{1}{8k} & \text{if } |\text{chil}(D_k)| + 1 \leq i \leq 2|\text{chil}(D_k)| \end{cases} \tag{A.8}$$

In the above equation, we have made use of the fact that $|\text{chil}(D_k)| \leq 8k$ because each node has at most **8** children. Also note that the probability of choosing any particular $b_i^{(k)}$ is $\frac{1}{|\text{chil}(D_k)|}$.

We now choose a node $\tilde{b}_*^{(k)}$ from the sequence $\widetilde{\text{chil}}(D_k)$ as follows. Let $\tilde{b}_*^{(k)}$ be the first node in the sequence $\widetilde{\text{chil}}(D_k)$ such that there is an open ball in $r_{x,y} \bigcap \text{support}(\psi_{\tilde{b}_*^{(k)}})$.

So $\tilde{b}_*^{(k)} = \tilde{b}_i^{(k)}$ means that there is an open ball in $r_{x,y} \bigcap \text{support}(\psi_{\tilde{b}_i^{(k)}})$ and there is no open ball in any of $r_{x,y} \bigcap \text{support}(\psi_{\tilde{b}_1^{(k)}}), \cdots, r_{x,y} \bigcap \text{support}(\psi_{\tilde{b}_i^{(k)}})$.

From the definition of $\tilde{b}_i^{(k)}$ and the probability assignment, it follows that

$$P(\tilde{b}_*^{(k)}) = \frac{1}{8k} \tag{A.9}$$

We now define various events whose probabilites we will evaluate. Let

$$Q_k = \{\exists \text{ an open ball in } r \bigcap \text{support}(\psi_{d_k})\} \tag{A.10}$$

$$S_k = \{\tilde{b}_*^{(k)} \text{ is picked}) \tag{A.11}$$

It is clear from the definition of $\tilde{b}_*^{(k)}$ that

$$\{S_k \text{ happens}\} \subset \{Q_k \text{ happens}), \quad \forall k \tag{A.12}$$

Therefore, we have

$$\{S_k \text{ infinitely often}) \subset \{Q_k \text{ infinitely often}) \tag{A.13}$$

Therefore, if we prove that the probability $P(\{S_k \text{ infinitely often})) = 1$, then we're done.

We now show that the S_k's are independent. First we see that for any k,

$$\begin{aligned} P(S_k/S_{k-1}, \cdots, S_1) &= P(S_k) \\ &= \frac{1}{8k} \\ &= P(S_k/S_{i_1}, S_{i_2}, \cdots), \quad 1 \leq i_1, i_2, \cdots < k \end{aligned} \tag{A.14}$$

Since this is true for $k = 1$ and k = 2, the independence of the S_k's follows from induction.

So S_k is an sequence of independent events and

$$\sum_{k=1}^{\infty} P(S_k) = \sum_{k=1}^{\infty} \frac{1}{8k}$$

which diverges. It therefore follows from the Borel Cantelli Lemma that

$$P(\limsup_{k} S_k) = 1$$

from which we can conclude that $P(\{S_k \text{infinitely often})) = 1$. This concludes the proof. ■

APPENDIX B: EXPERIMENTAL RESULTS FOR VARIOUS IMAGES AND WAVELETS

In this appendix, we present experimental results for each of the 20 images analysed with each of the four wavelets. The 20 images are shown in figure (1.1). The four wavelets used were the Haar wavelet, Daubechies D4, spline of order 2, and the spline-variant. These were described in chapter 2.

We show the following plots for each image-wavelet combination:

a. Plots of magnitudes versus Sort-Index (which looks like $\frac{1}{x^\alpha}$).

b. Log-log plot of magnitude versus Sort-Index (which looks like a straight line).

c. Plot of magnitudes versus TAS-Index (which looks like $\frac{1}{x^\alpha}$ with some points above the curve).

d. Plot of TAS-Index versus Sort-Index (from which we derive γ).

Of the 20 images in the database of figure (1.1), the last three are synthetic images ("Oval", “Oval+Noise” and "Noise"). For these images, the ;log-log plot does not appear like a straight line, so the form $\frac{1}{x^\alpha}$ is not a good approximation of the plot of magnitudes versus Sort-Index. Further, for some high-detail images (like the "Baboon" image), the log-log plot appears somewhat like a straight line but also contains higher order terms. Thus, a more complicated form (than $\frac{1}{x^\alpha}$) may be required to get a good fit to the curve of magnitudes versus Sort-Inclex.

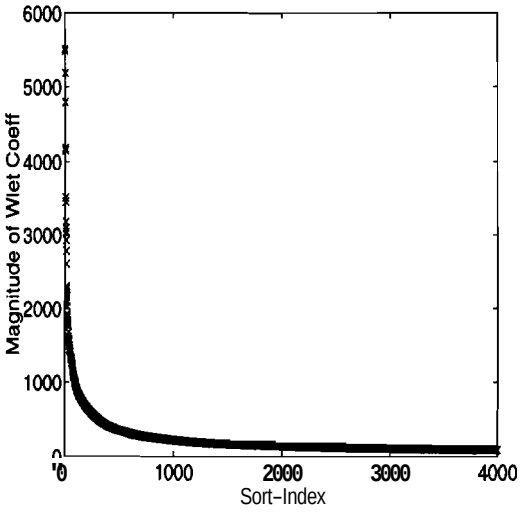
Note, however, that the image compression algorithm does not, in any way, depend upon the form of the curve of magnitudes versus Sort-Index. The only requirement is that the curve decay rapidly initially. This requirement is met by all the images analysed with each of the wavelets used.

The main point of the figures in this appendix is that the TAS algorithm closely approximates a sort by magnitude. This can be seen as follows. The figures (B.1d -

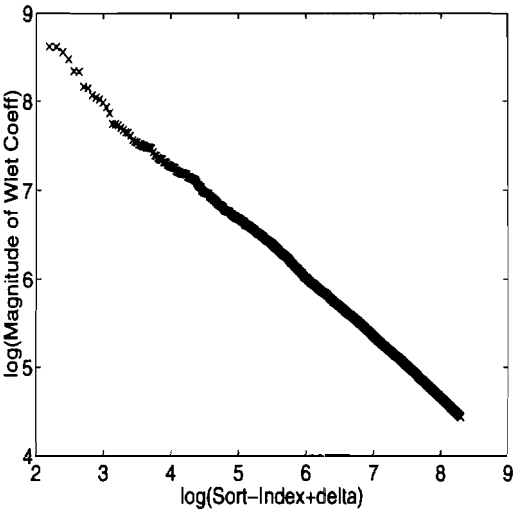
B.80d) show that there exists a number γ for which equation (3.3) is satisfied. This immediately leads to the conclusion that equation (3.6) is valid. Note that even if the form $\frac{1}{x^\alpha}$ is not strictly valid, the argument leading to equation (3.6) nevertheless guarantees that the error in our approximation decreases rapidly. A rapid decay of this error is the main ingredient that makes our image compression algorithm work.

Another point of the figures in this appendix is that the form $\frac{1}{x^\alpha}$ is indeed a reasonable approximation for the plot of first 4000 sorted magnitudes for a wide variety of images. This can be seen from figures (a,b) of the entire set of images in this appendix. Also note that if the log-log plot is plotted for the entire set of $512^2 = 262144$ coefficients, we will find that in the latter part of the graph, the coefficients are even smaller than the value predicted by the $\frac{1}{x^\alpha}$ curve. Since the coefficients that are eliminated from the compressed representation are even smaller than predicted, this poses no problem for the image compression algorithm.

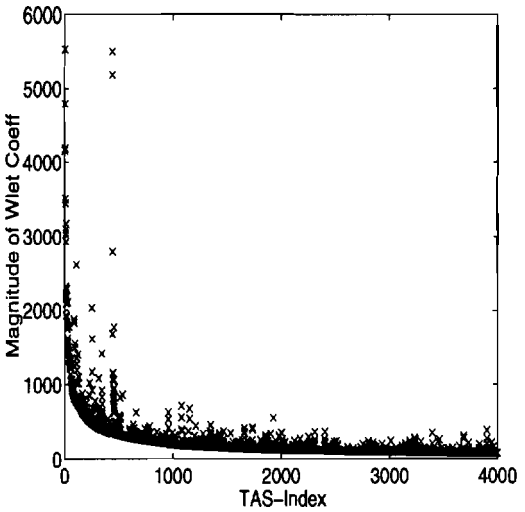
It is to be noted is that in figures (B.1d - B.76d), we have not shown data points for which the TAS-Index is greater than the largest value of Sort-Index shown (i.e. 4000). This is because the purpose of these graphs is to illustrate the fact that (3.3) gives an empirical lower bound for the TAS-Index. Note that (3.3) gives no information on how large the TAS-Index may become. Therefore, we can verify (3.3) using these figures even if we ignore the very large values that distort the scale of the graph.



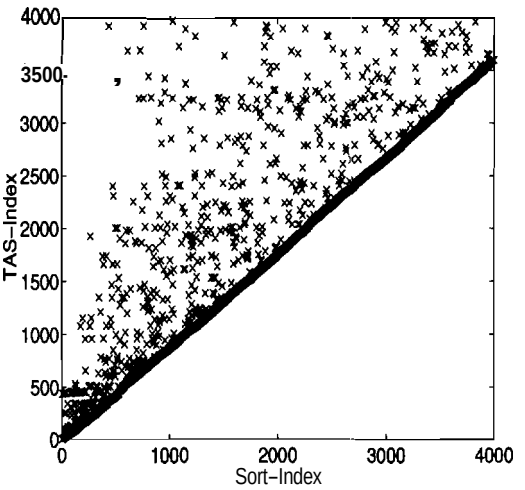
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

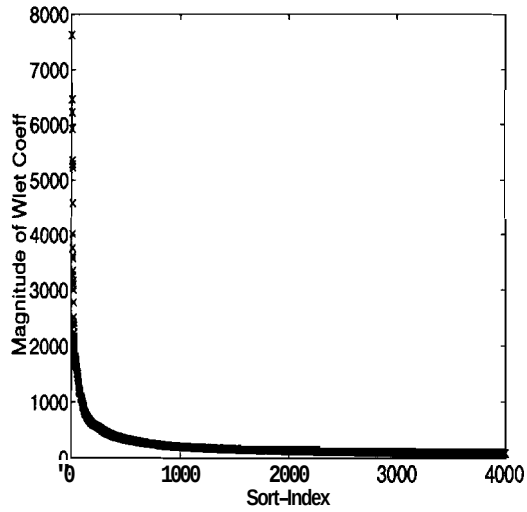


(c) Magnitudes vs TAS-Index

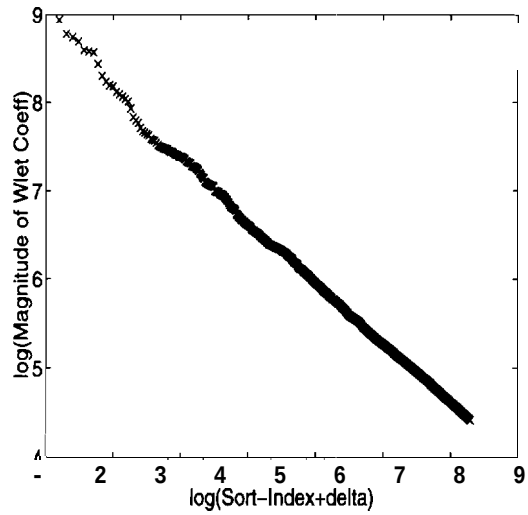


(d) TAS-Index vs Sort-Index

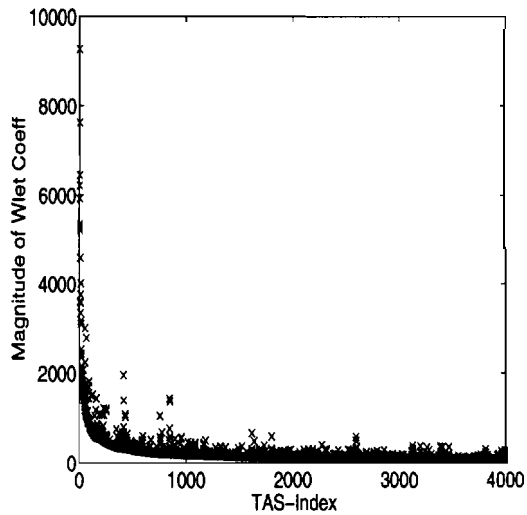
Fig. B.1. "Lenna" image with Haar Wavelet.



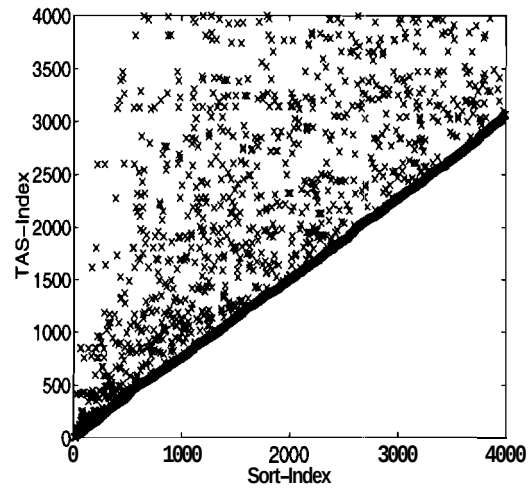
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

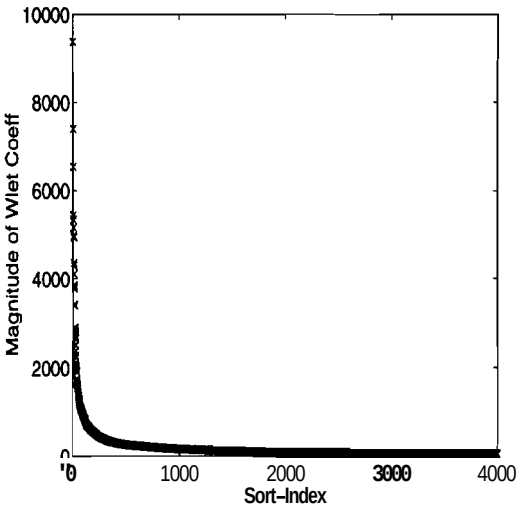


(c) Magnitudes vs TAS-Index

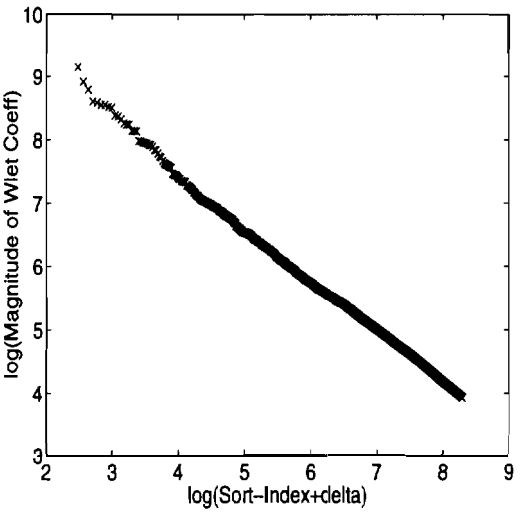


(d) TAS-Index vs Sort-Index

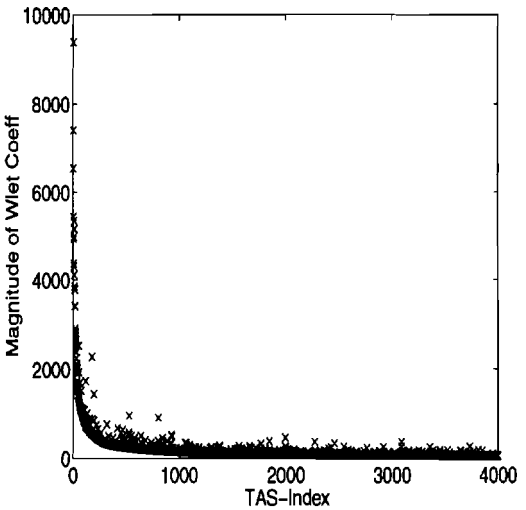
Fig. B.2. "Lenna" image with Daubechies D4 Wavelet.



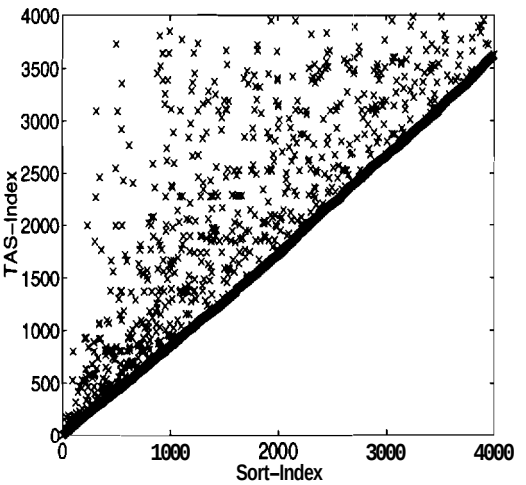
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

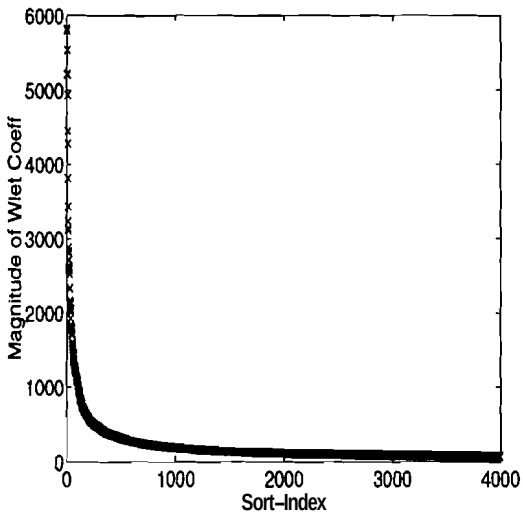


(c) Magnitudes vs TAS-Index

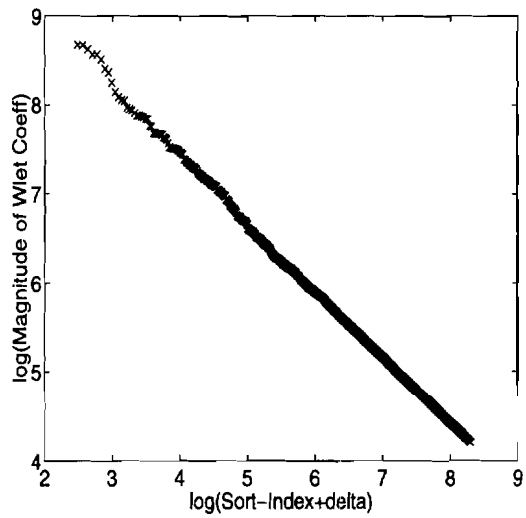


(d) TAS-Index vs Sort-Index

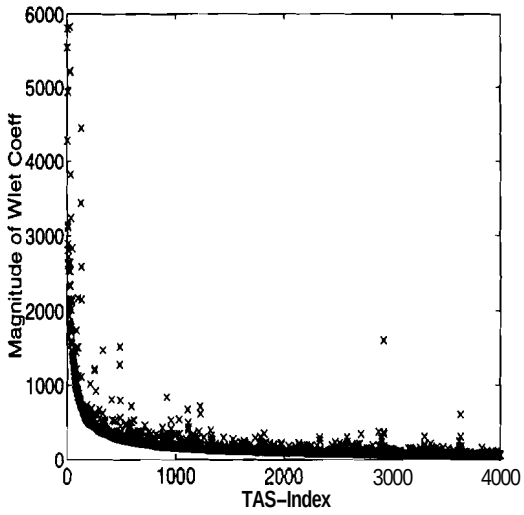
Fig. **B.3.** "Lenna" image with Spline-2 Wavelet.



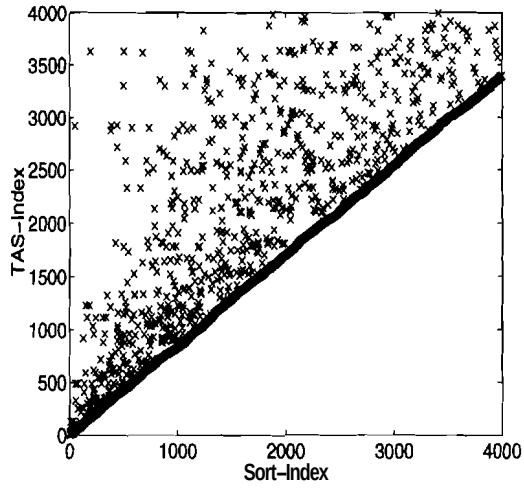
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

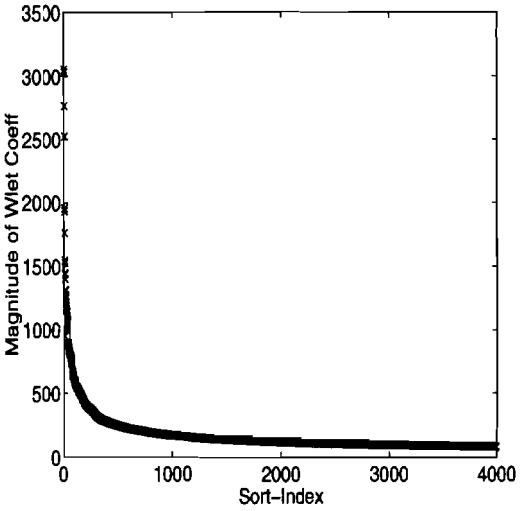


(c) Magnitudes vs TAS-Index

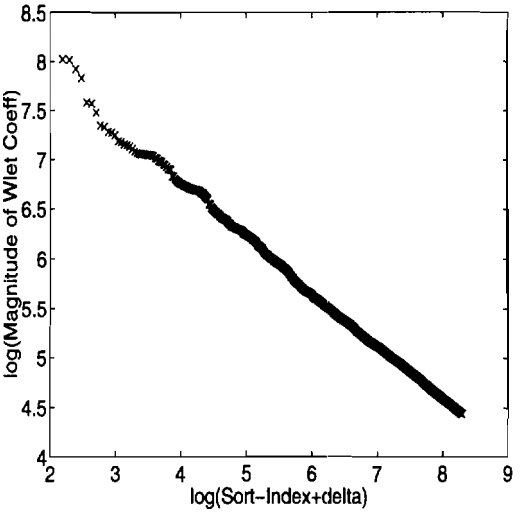


(d) TAS-Index vs Sort-Index

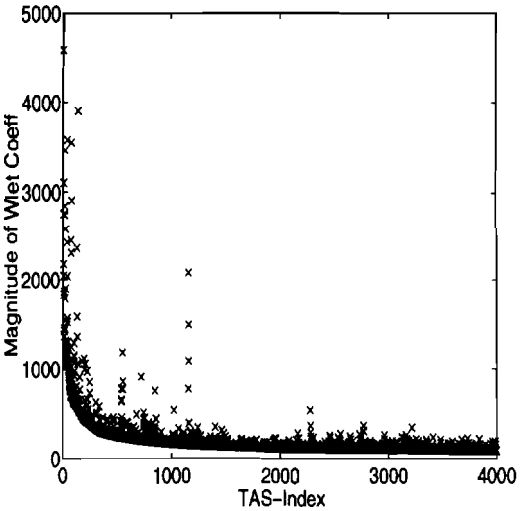
Fig. B.4. "Lenna" image with spline-variant Wavelet.



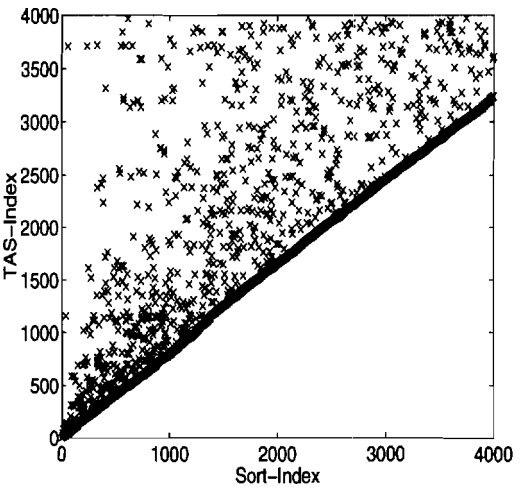
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

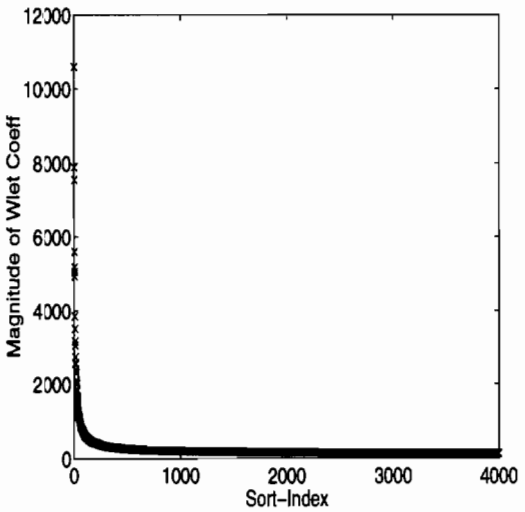


(c) Magnitudes vs TAS-Index

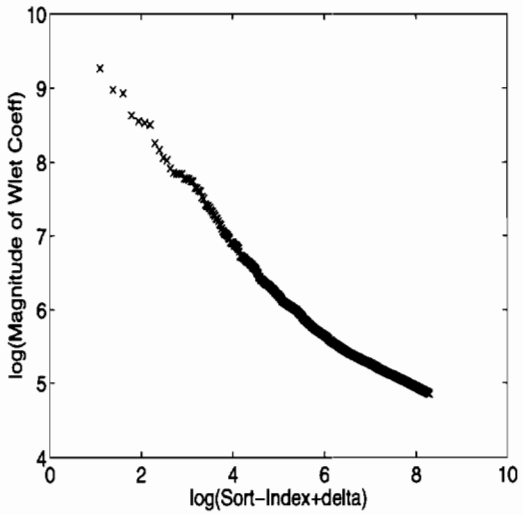


(d) TAS-Index vs Sort-Index

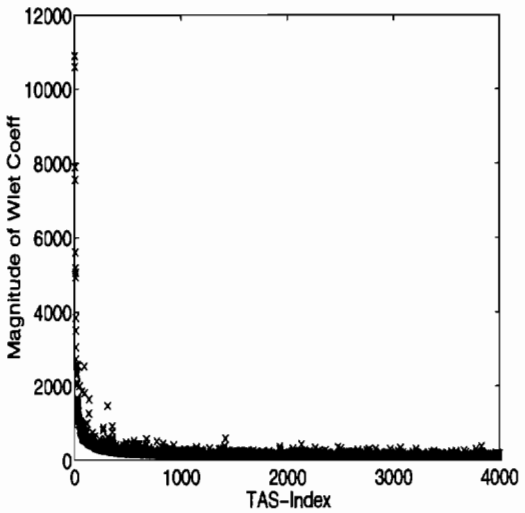
Fig. B.5. High-detail image: "Baboon" image with Haar Wavelet.



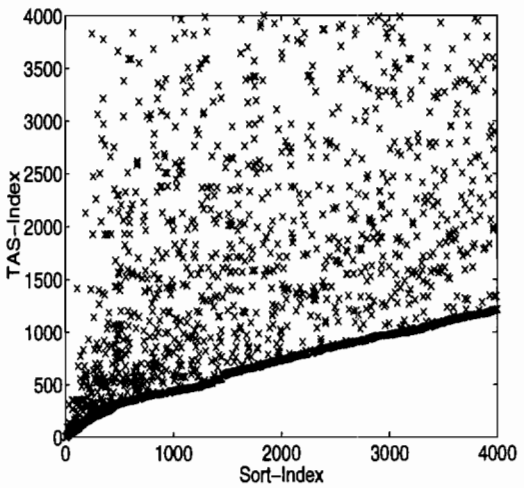
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

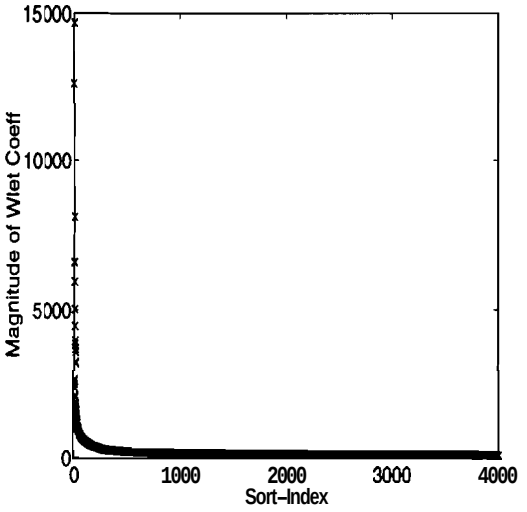


(c) Magnitudes vs TAS-Index

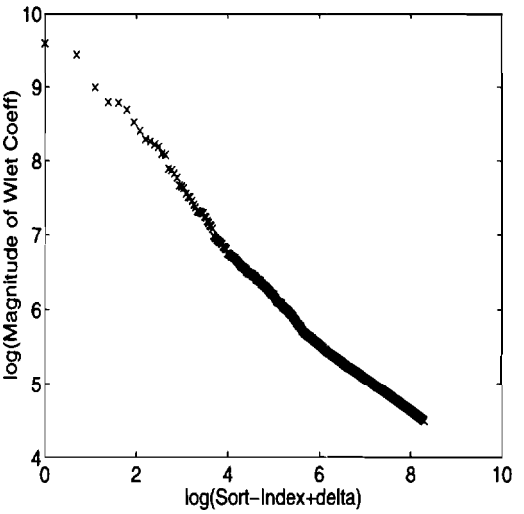


(d) TAS-Index vs Sort-Index

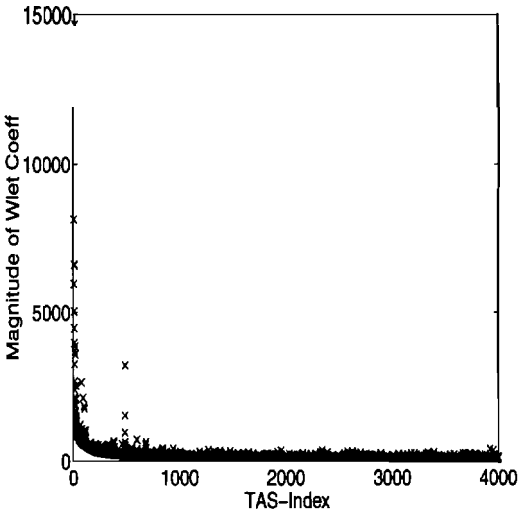
Fig. B.6. High-detail image: "Baboon" image with Daubechies D4 Wavelet.



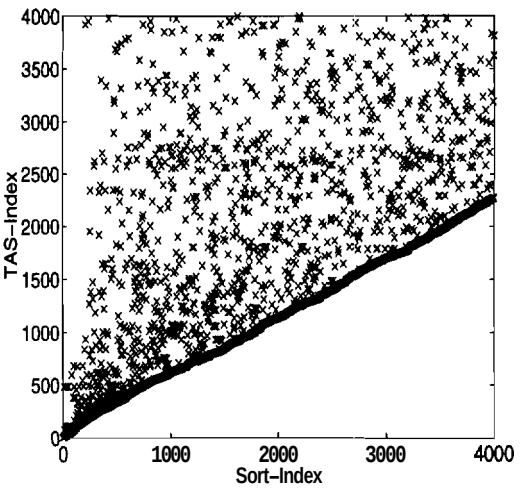
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

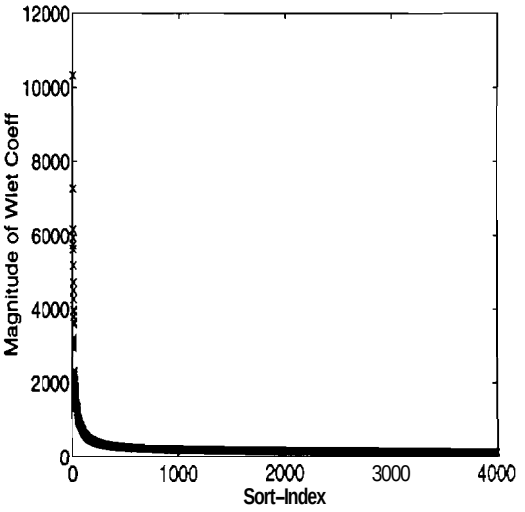


(c) Magnitudes vs TAS-Index

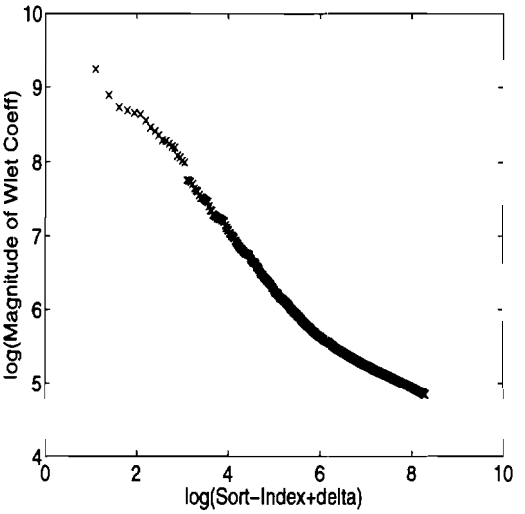


(d) TAS-Index vs Sort-Index

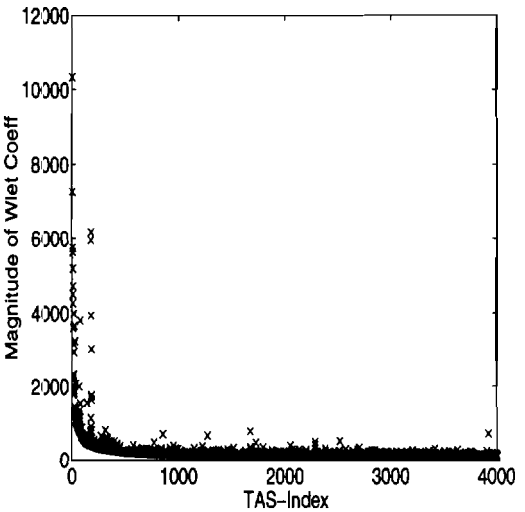
Fig. B.7. High-detail image: "Baboon" image with Spline-2 Viavelet.



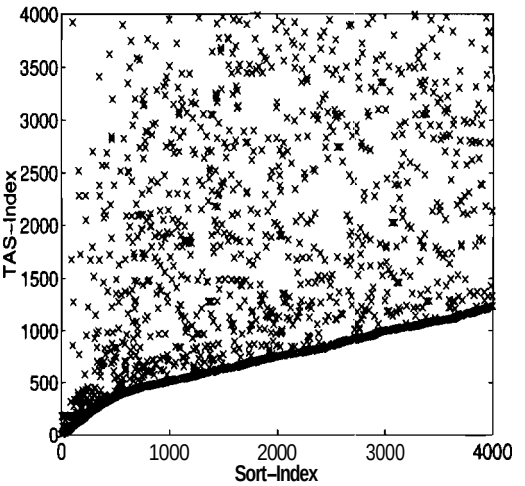
(a) Magnitudes vs Sort-Index



(b) log-log plot of **(a)**

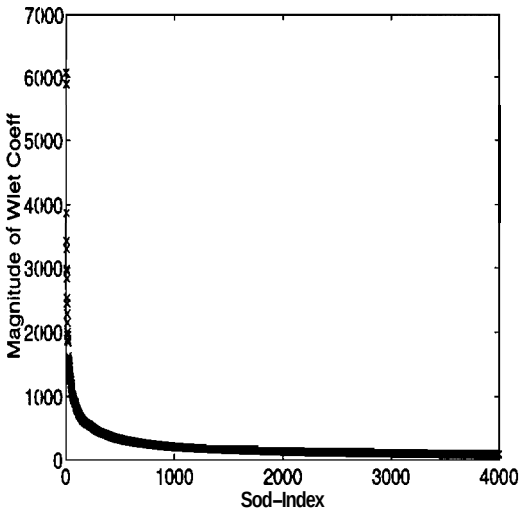


(c) Magnitudes vs TAS-Index

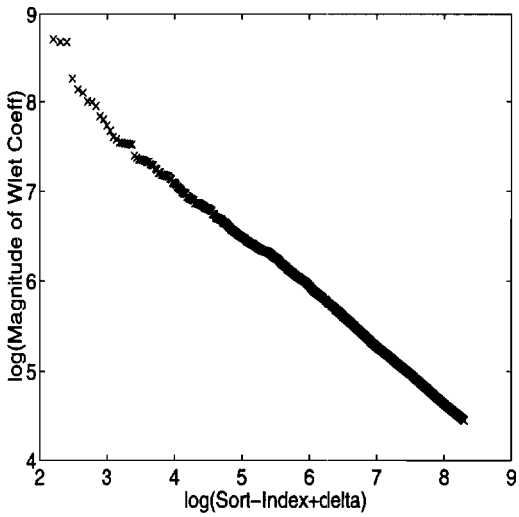


(d) TAS-Index vs Sort-Index

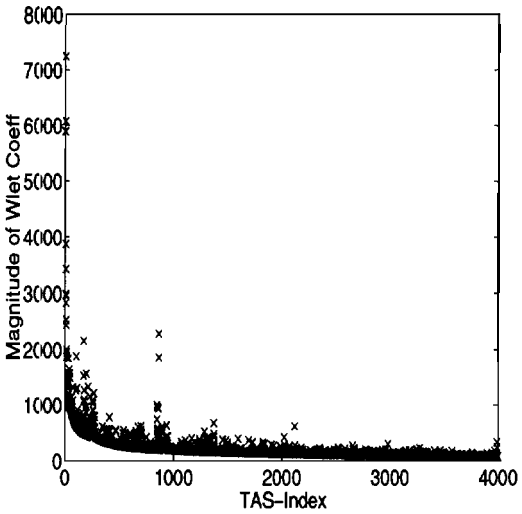
Fig. B.8. High-detail image: "Baboon" image with spline-variant Wavelet.



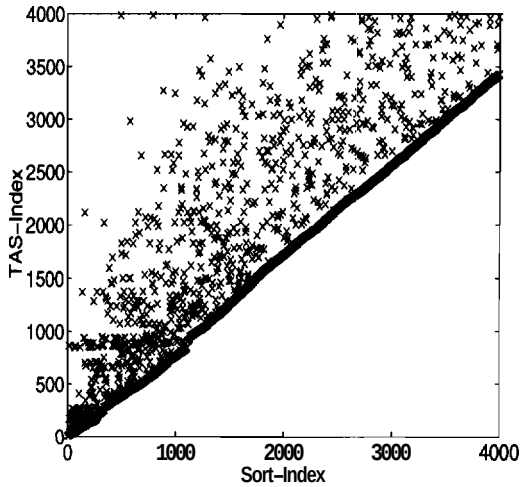
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

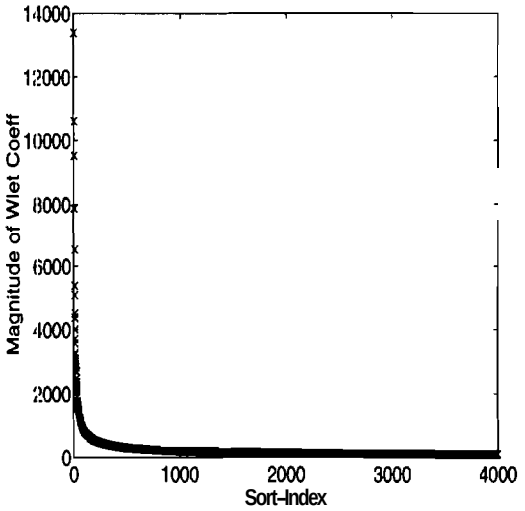


(c) Magnitudes vs TAS-Index

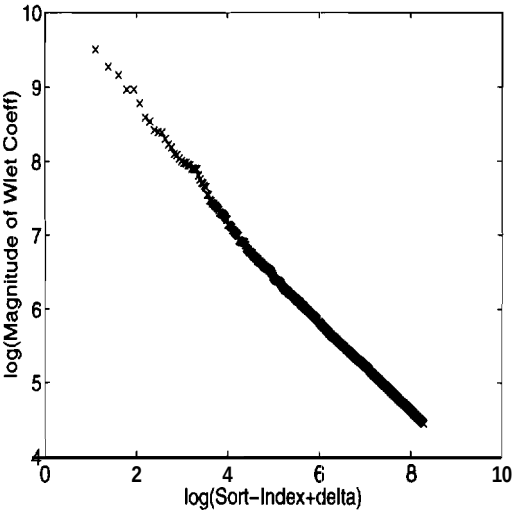


(d) TAS-Index vs Sort-Index

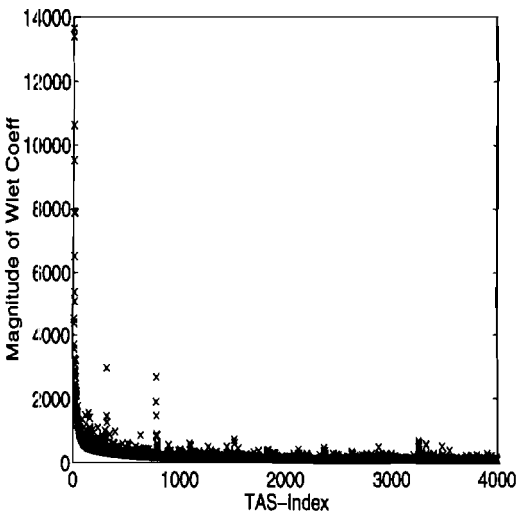
Fig. B.9. "Car" image with Haar Wavelet.



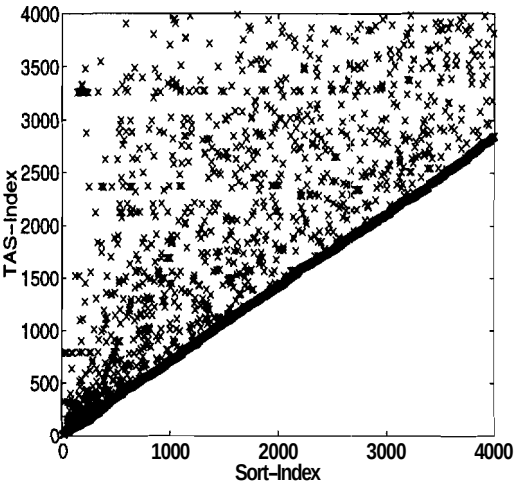
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

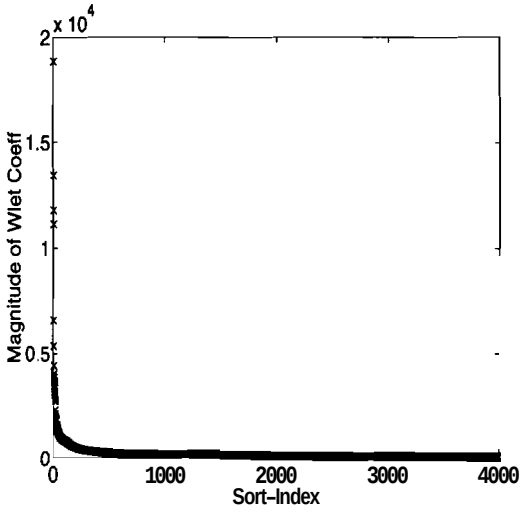


(c) Magnitudes vs TAS-Index

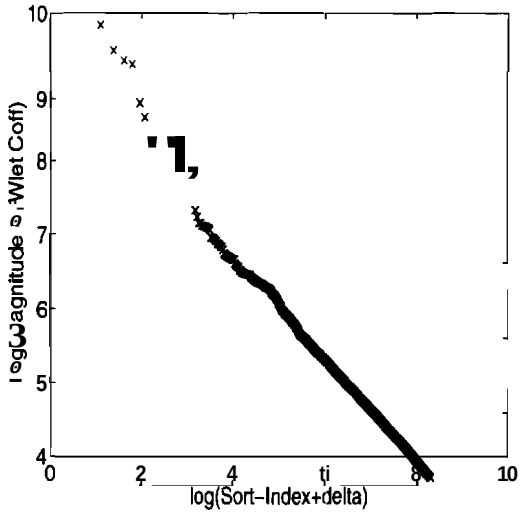


(d) TAS-Index vs Sort-Index

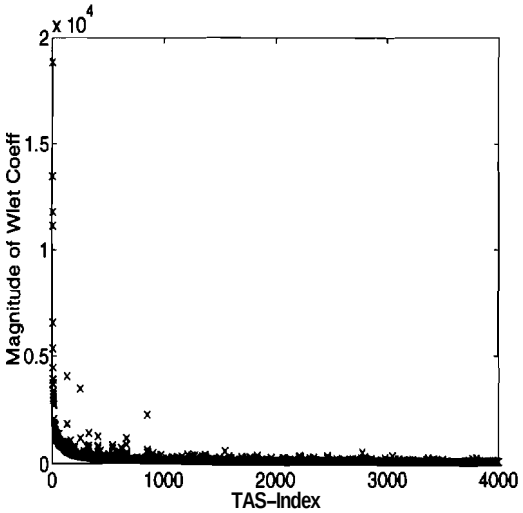
Fig. B.lO. "Car" image with Daubechies D4 Wavelet.



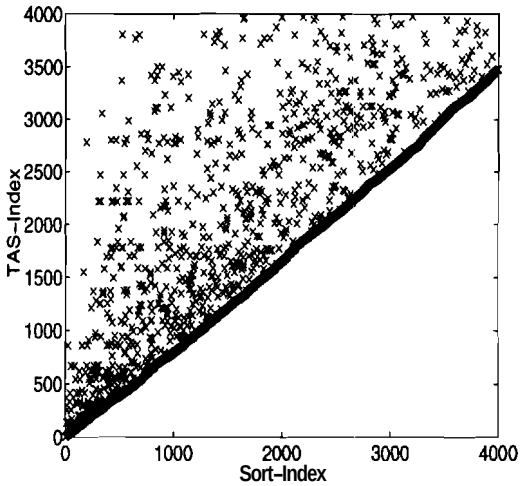
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

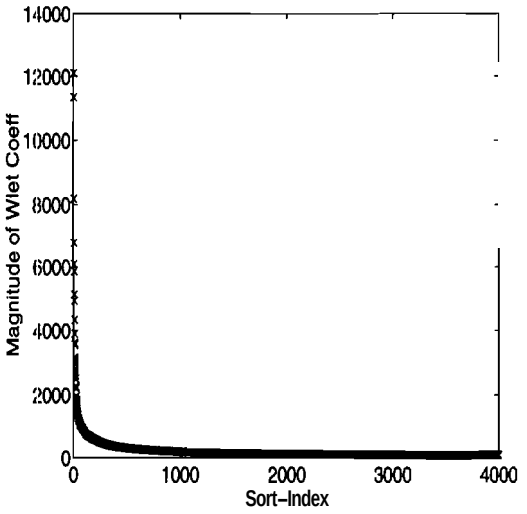


(c) Magnitudes vs TAS-Index

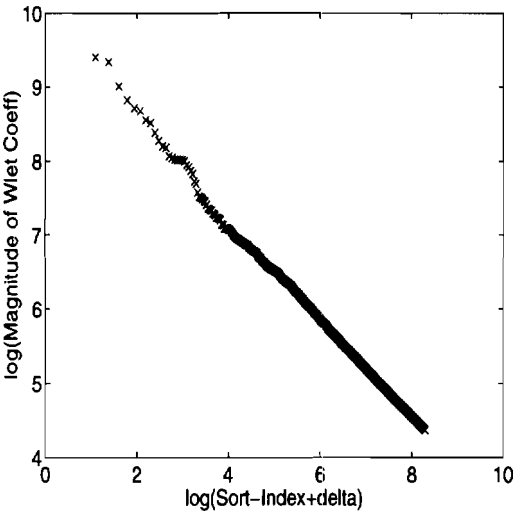


(d) TAS-Index vs Sort-.Index

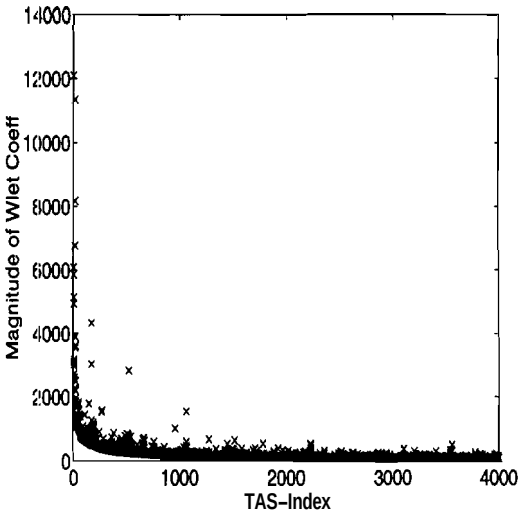
Fig. B.11. "Car" image with Spline-2 Wavelet.



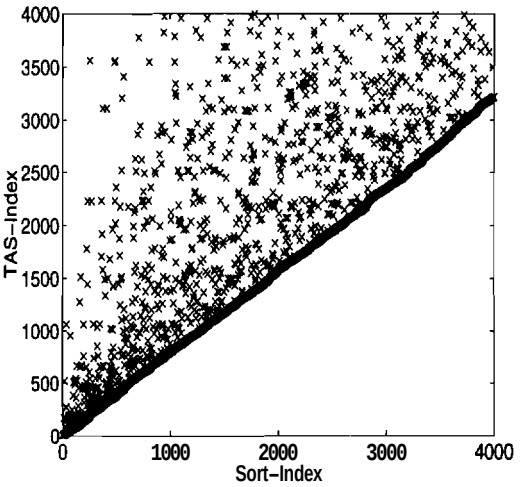
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

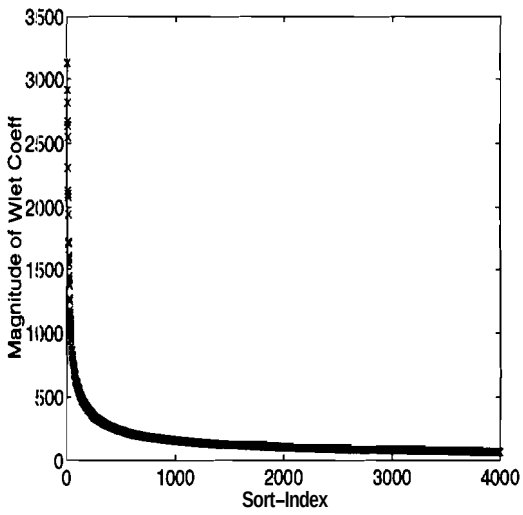


(c) Magnitudes vs TAS-Index

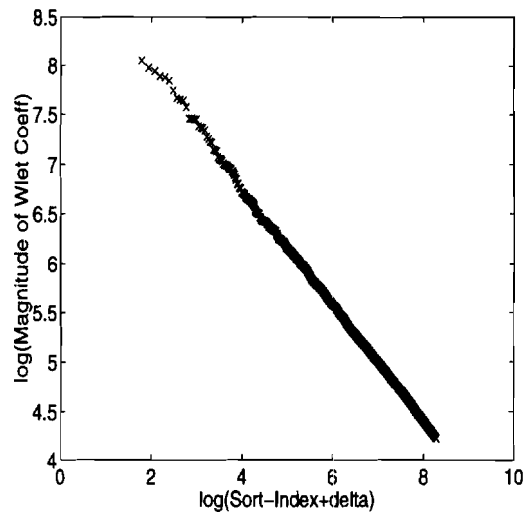


(d) TAS-Index vs Sort-Index

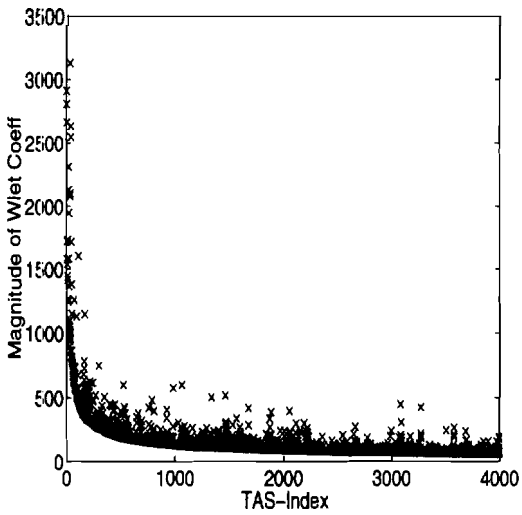
Fig. B.12. "Car" image with spline-variant Wavelet.



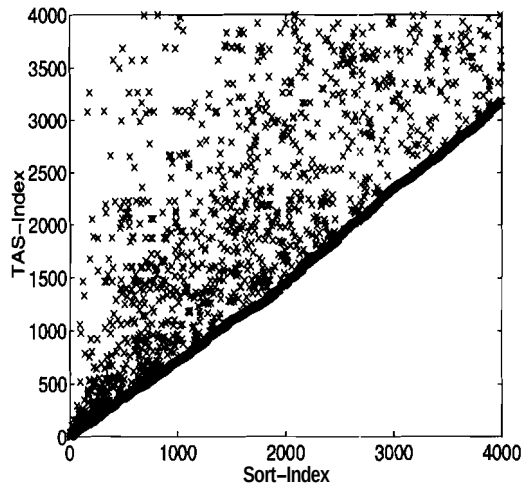
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

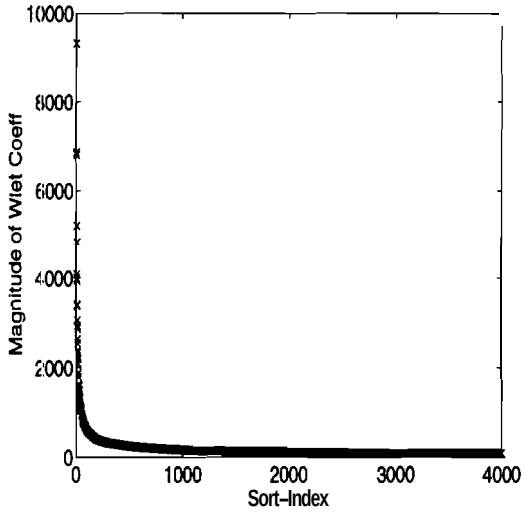


(c) Magnitudes vs TAS-Index

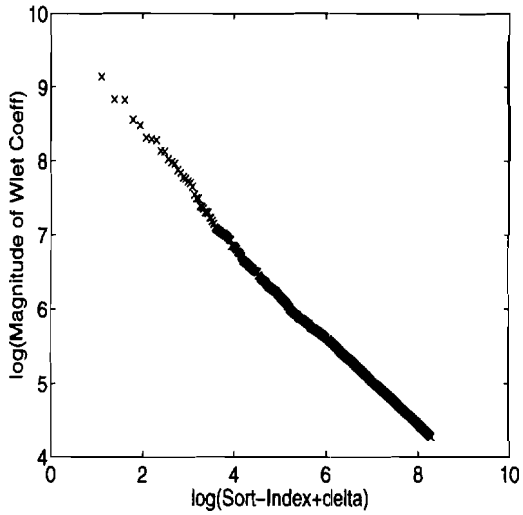


(d) TAS-Index vs Sort-Index

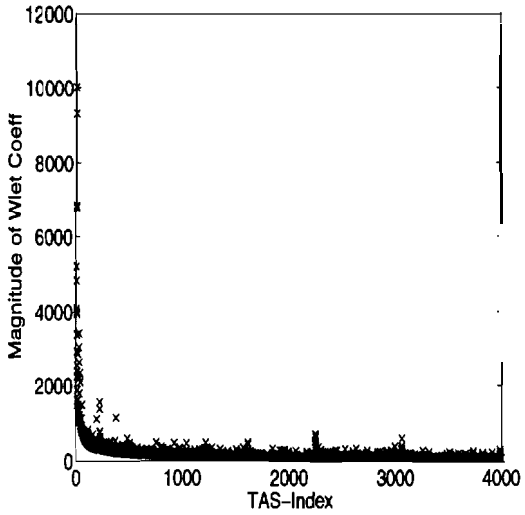
Fig. **B.13.** "Couple" image with Haar Wavelet.



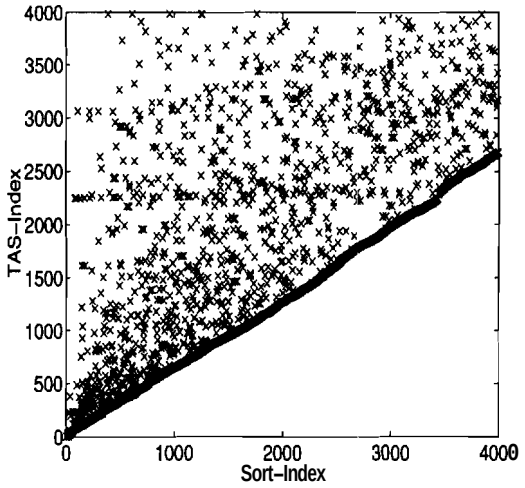
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

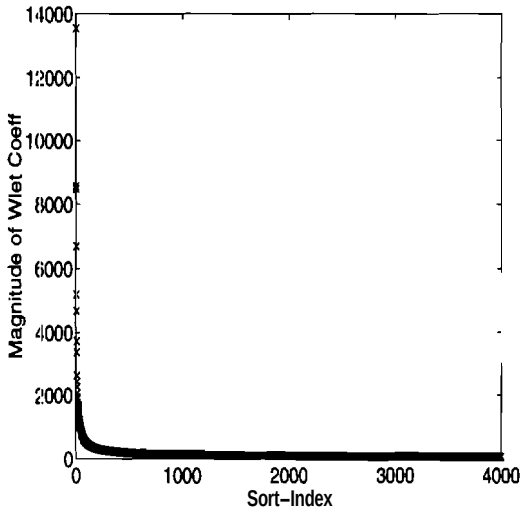


(c) Magnitudes **vs** TAS-Index

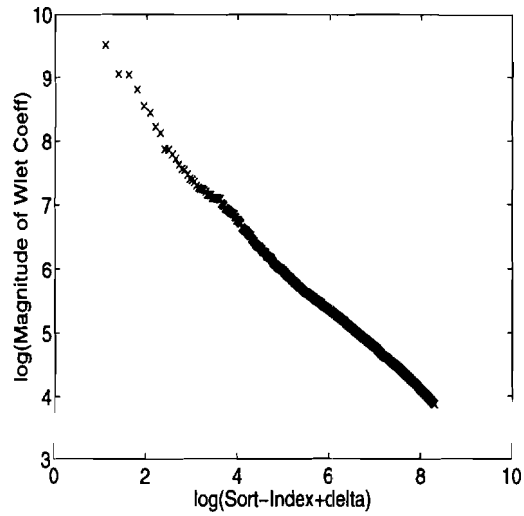


(d) TAS-Index vs Sort-Index

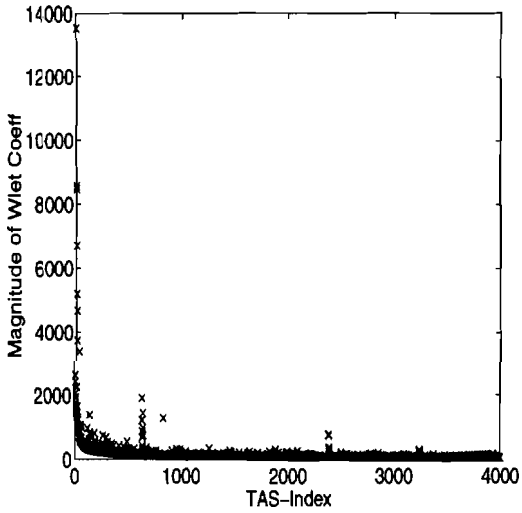
Fig. B.14. "Couple" image with Daubechies D4 Wavelet.



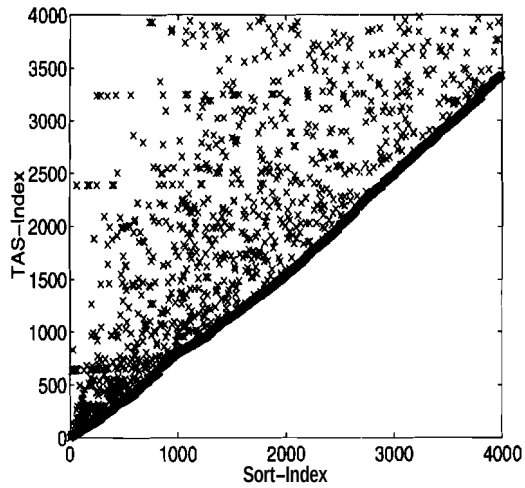
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

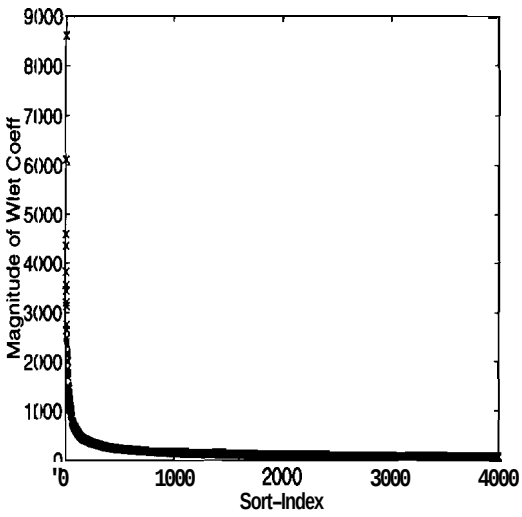


(c) Magnitudes vs TAS-Index

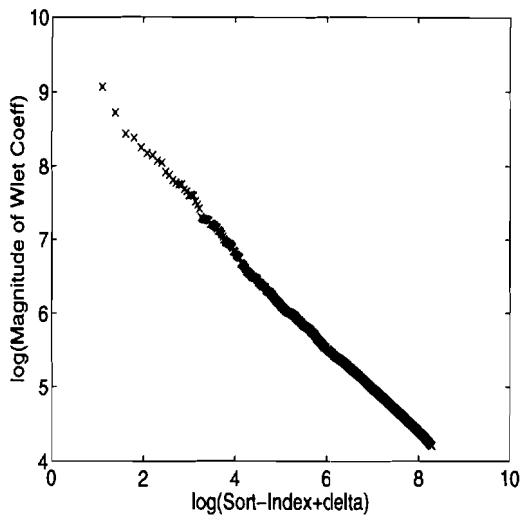


(d) TAS-Index vs Sort-Index

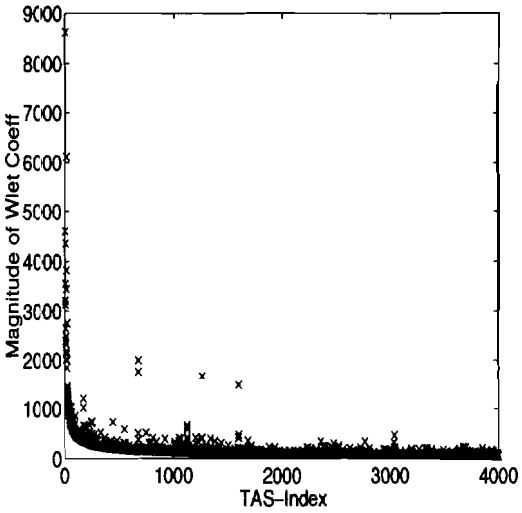
Fig. B.15. "Couple" image with Spline-2 Wavelet.



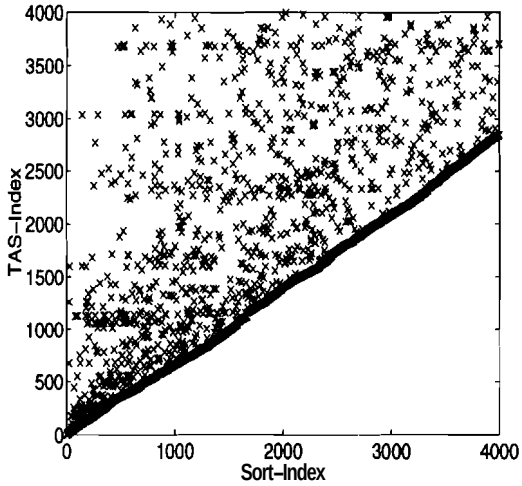
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

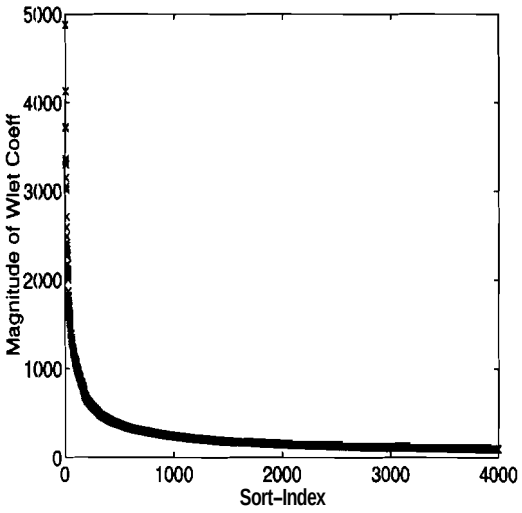


(c) Magnitudes vs TAS-Index

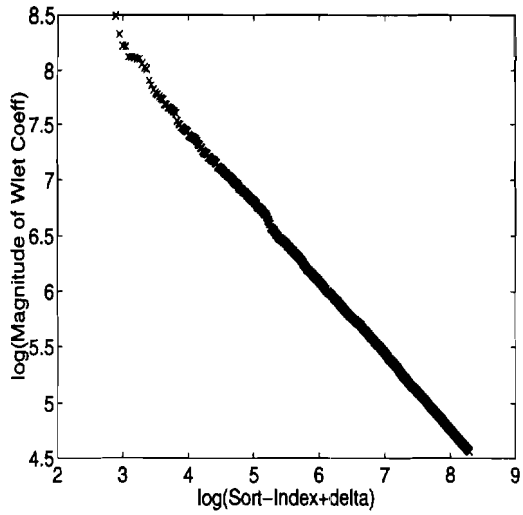


(d) TAS-Index vs Sort-Index

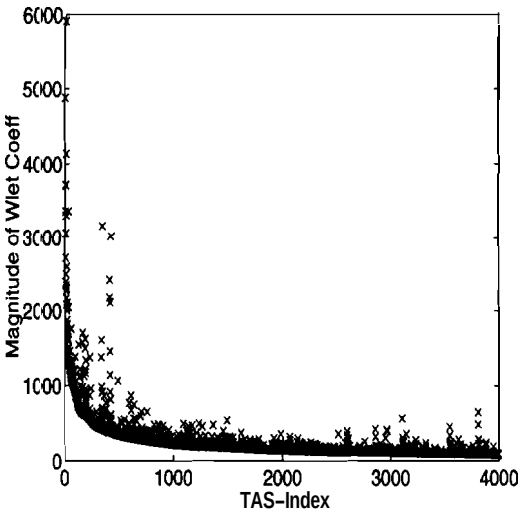
Fig. B.16. "Couple" image with spline-variant Wavelet.



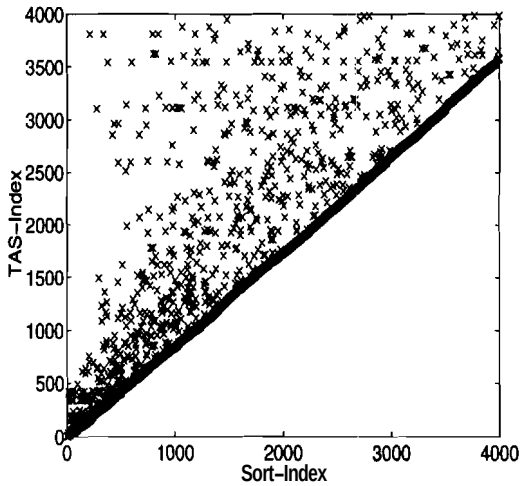
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

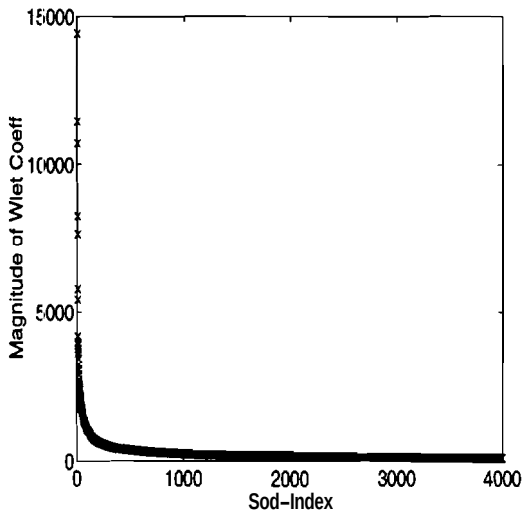


(c) Magnitudes vs TAS-Index

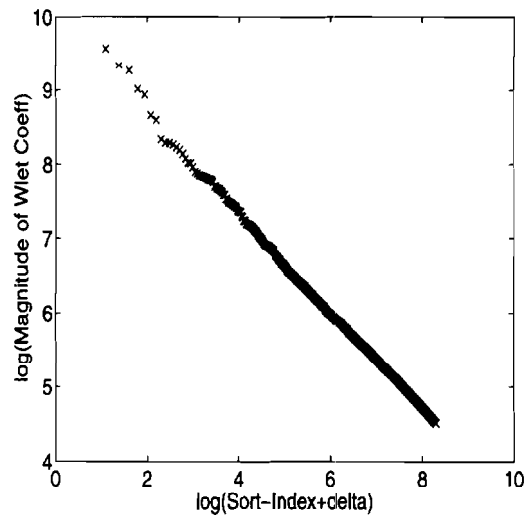


(d) TAS-Index vs Sort-Index

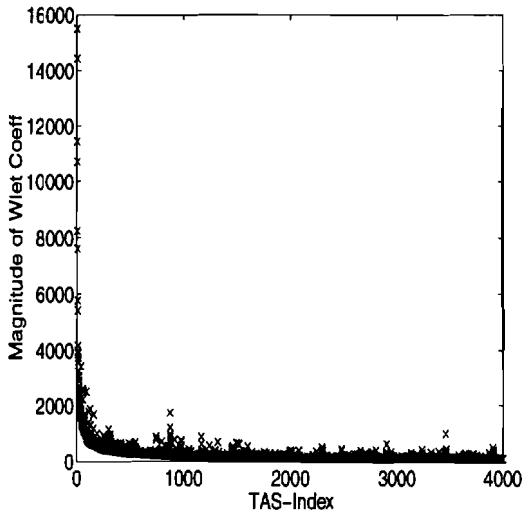
Fig. B.17. "F16" image with Haar Wavelet.



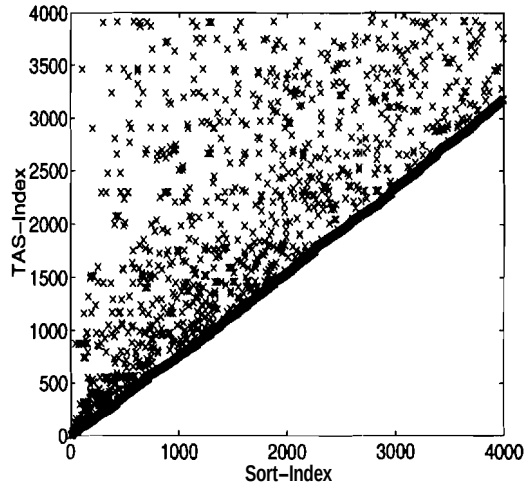
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

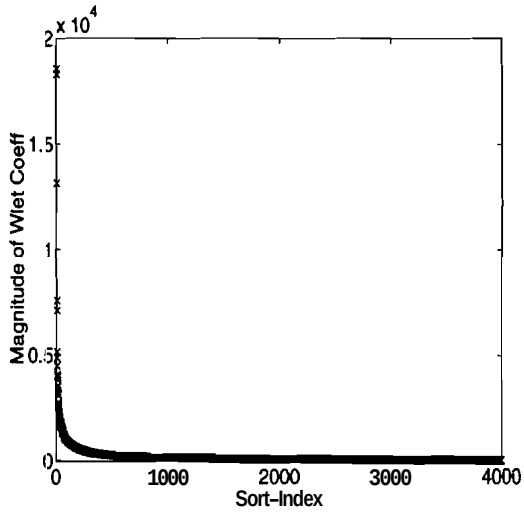


(c) Magnitudes vs TAS-Index

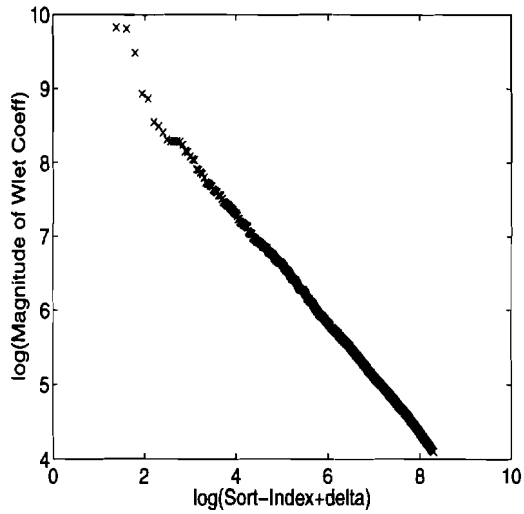


(d) TAS-Index vs Sort-Index

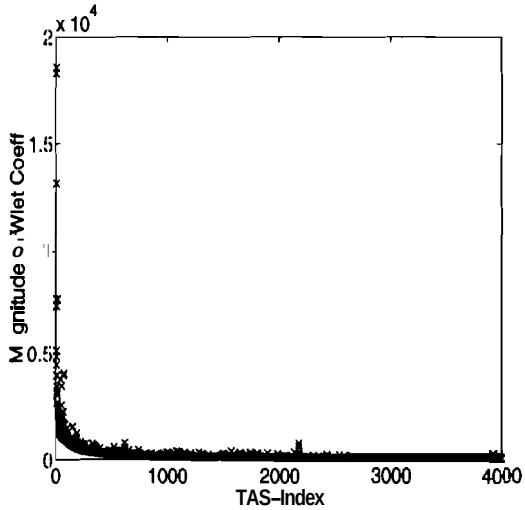
Fig. B.18. "F16" image with Daubechies D4 Wavelet.



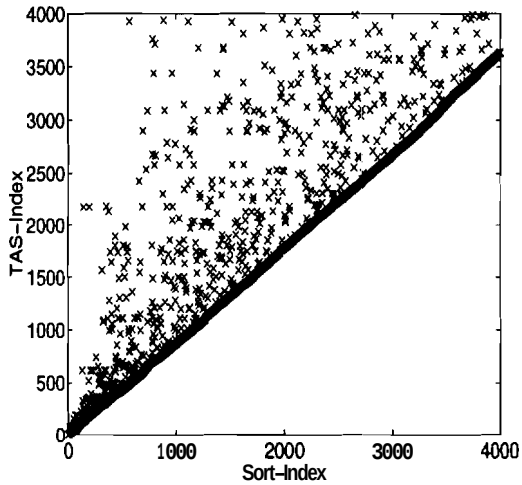
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

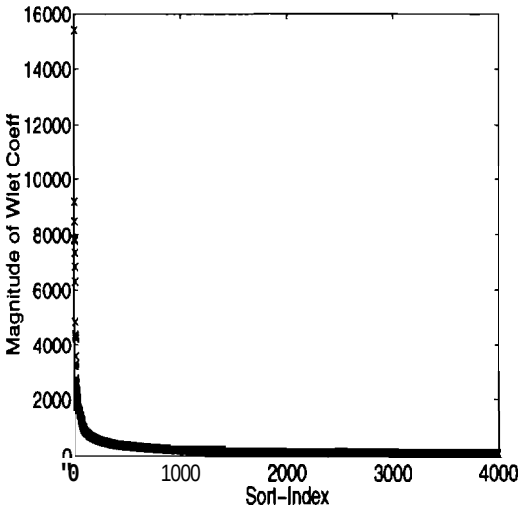


(c) Magnitudes vs TAS-Index

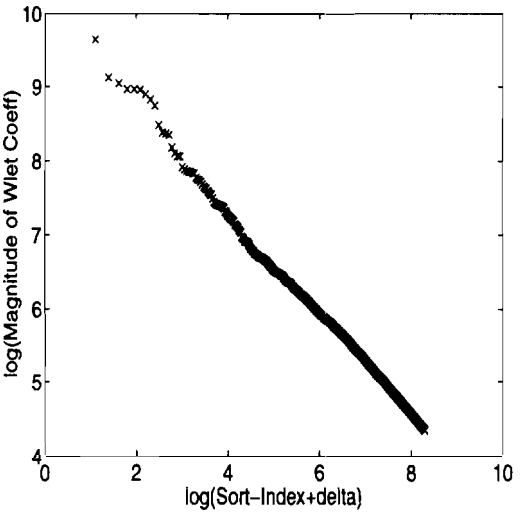


(d) TAS-Index vs Sort-Index

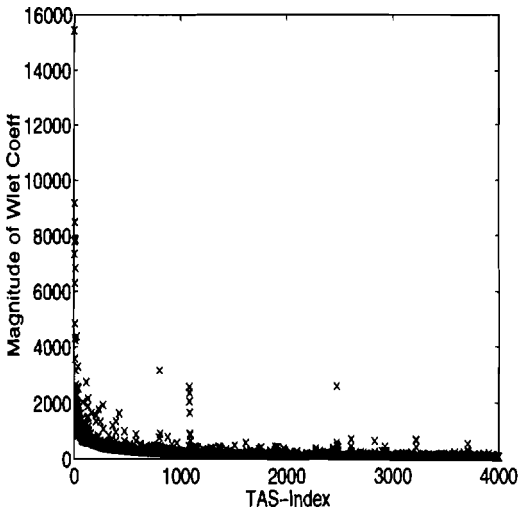
Fig. B.19. "F16" image with Spline-2 Wavelet.



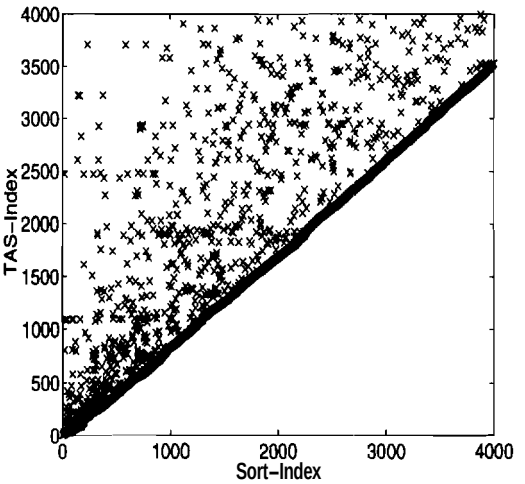
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

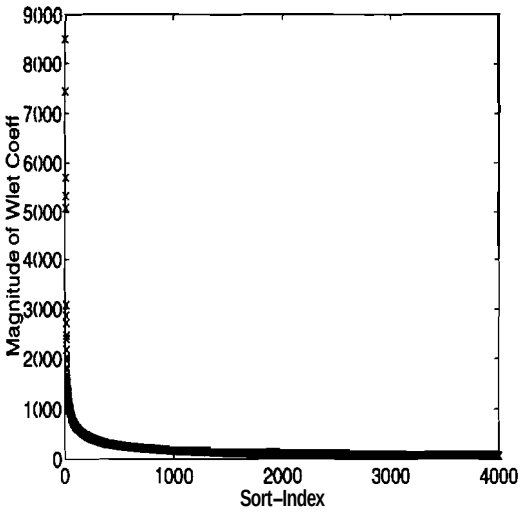


(c) Magnitudes vs TAS-Index

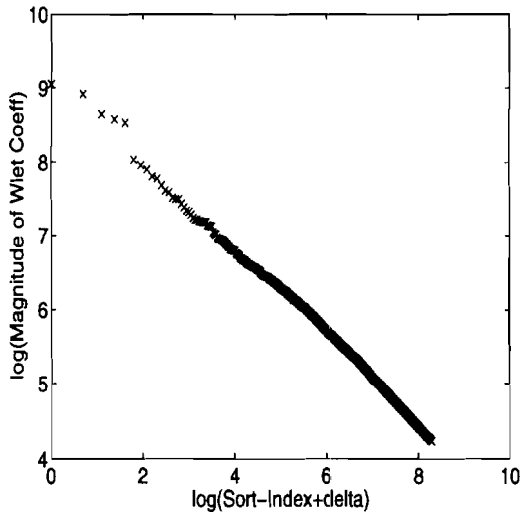


(d) TAS-Index vs Sort-Index

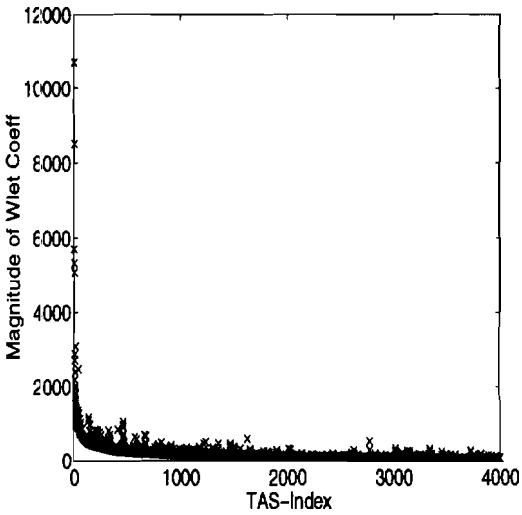
Fig. B.20. "F16" image with spline-variant Wavelet.



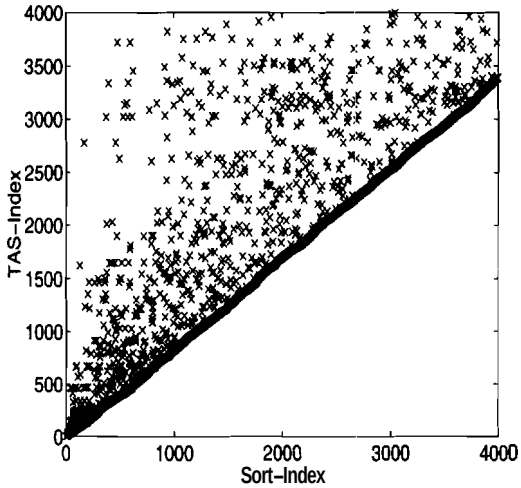
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

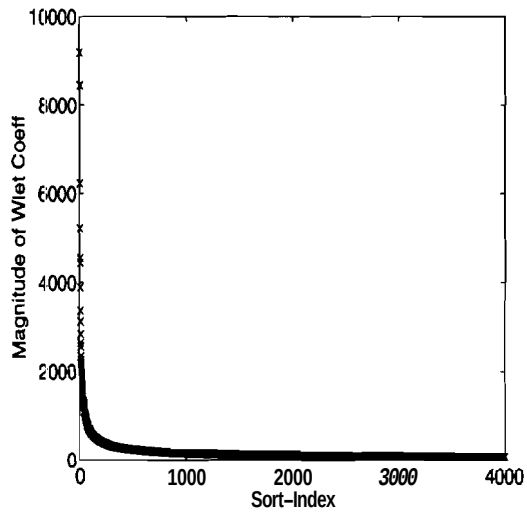


(c) Magnitudes vs TAS-Index

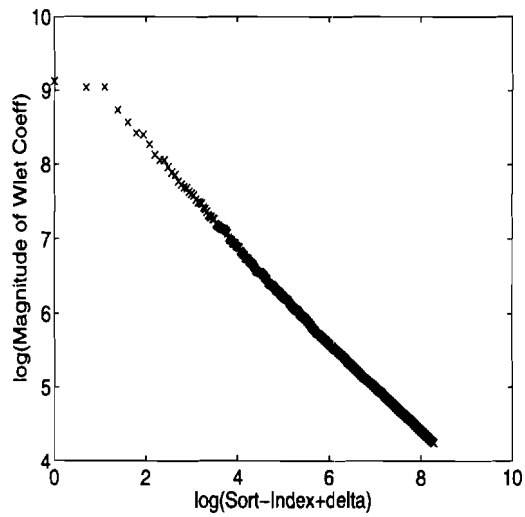


(d) TAS-Index vs Sort-Index

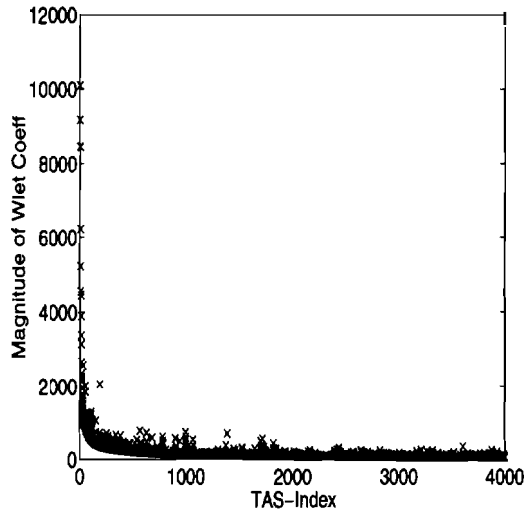
Fig. B.21. "Fields" image with Haar Wavelet.



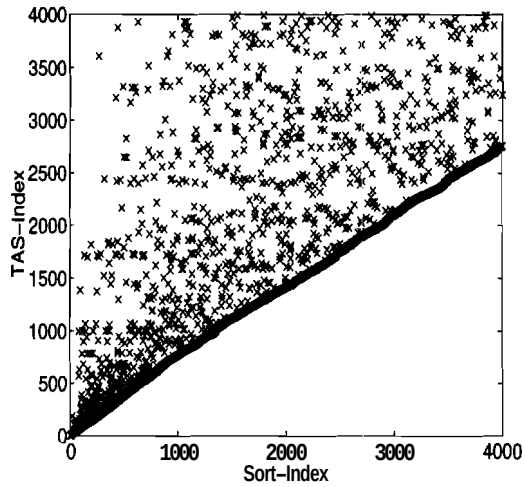
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

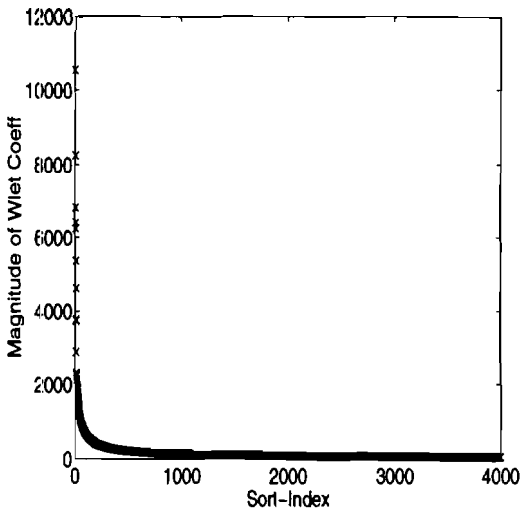


(c) Magnitudes vs TAS-Index

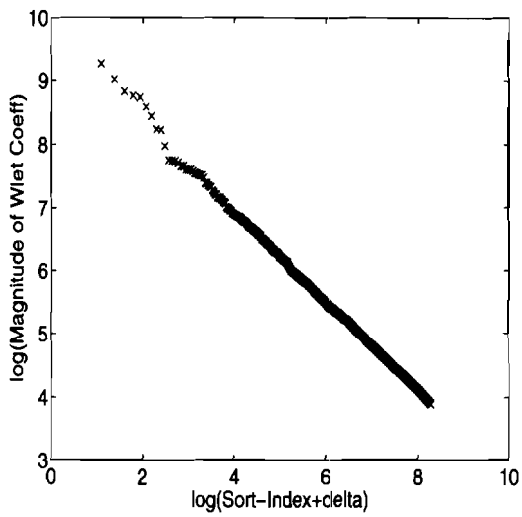


(d) TAS-Index vs Sort-Index

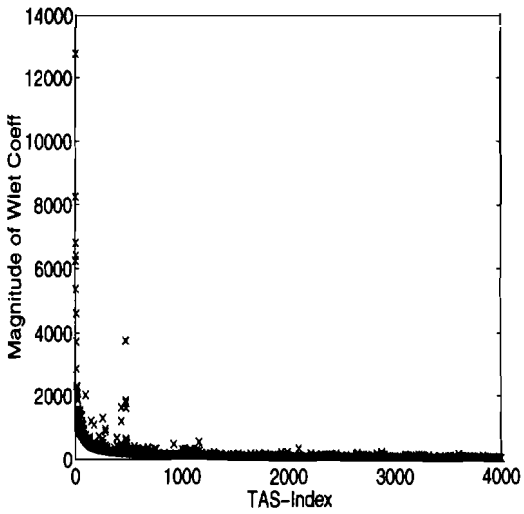
Fig. B.22. "Fields" image with Daubechies D4 Wavelet.



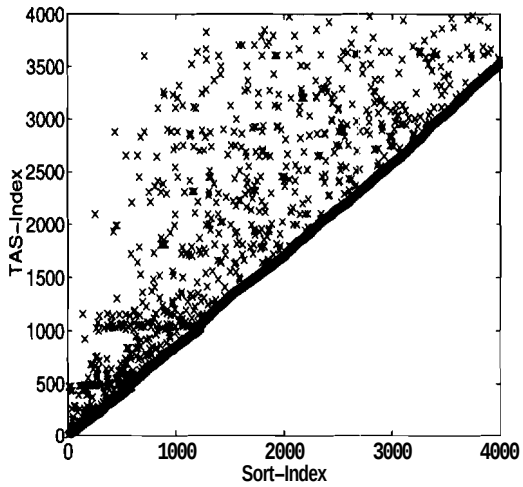
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

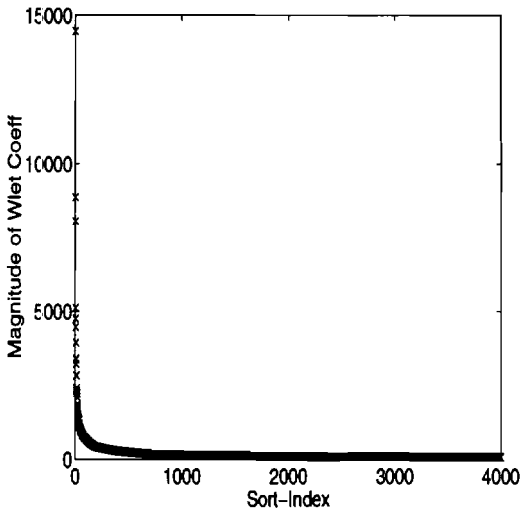


(c) Magnitudes vs TAS-Index

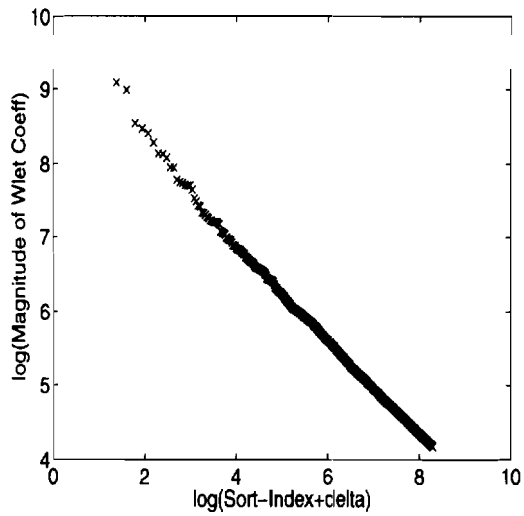


(d) TAS-Index vs Sort-Index

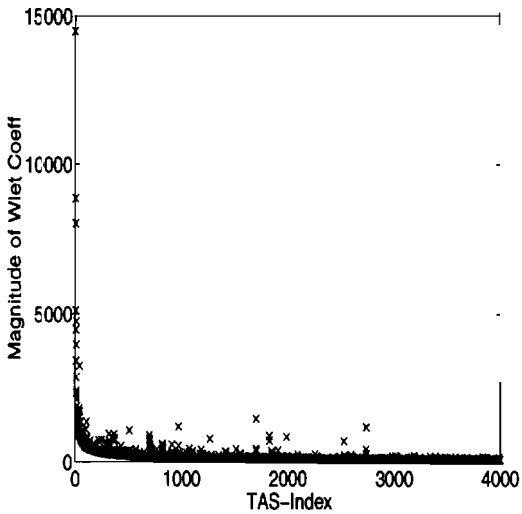
Fig. B.23. "Fields" image with Spline-2 Wavelet.



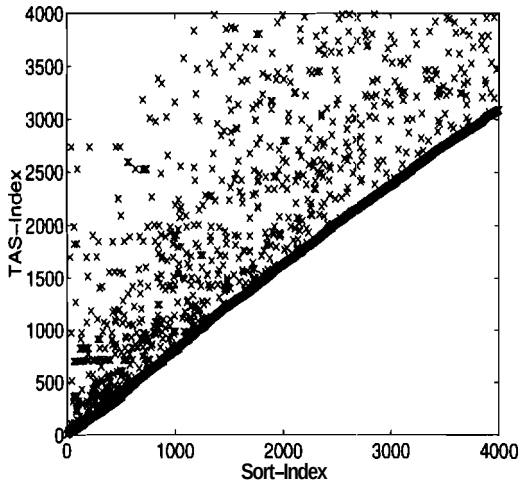
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

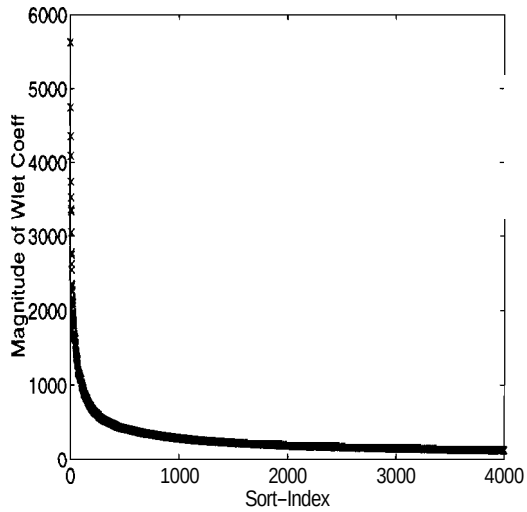


(c) Magnitudes vs TAS-Index

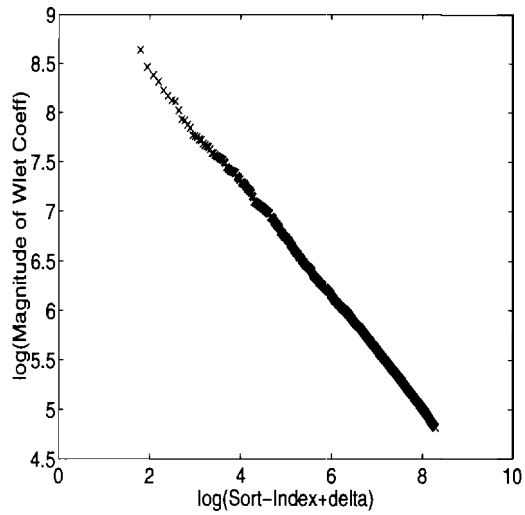


(d) TAS-Index vs Sort-Index

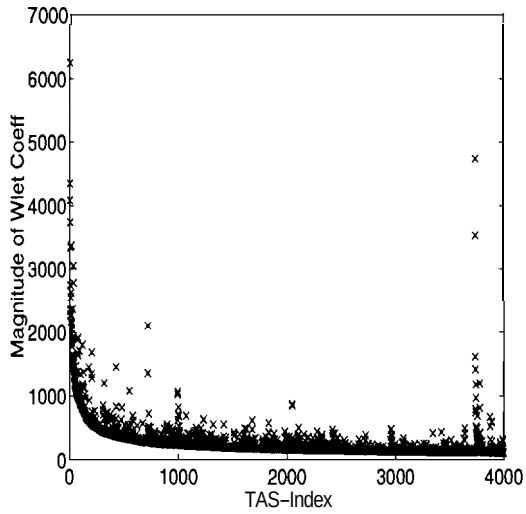
Fig. B.24. "Fields" image with spline-variant Wavelet.



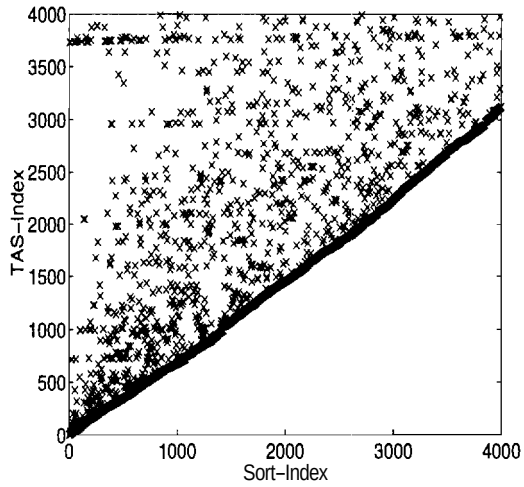
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

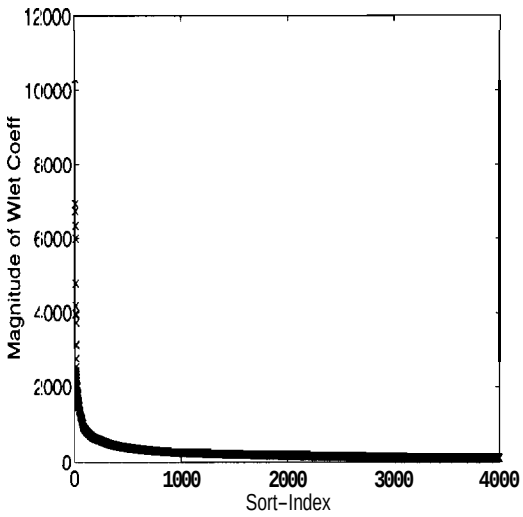


(c) Magnitudes vs TAS-Index

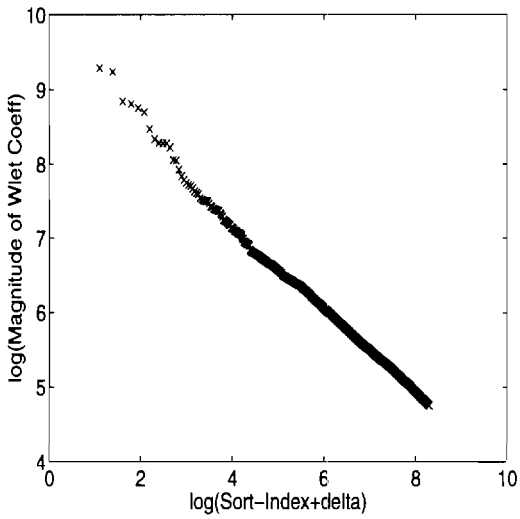


(d) TAS-Index vs Sort-Index

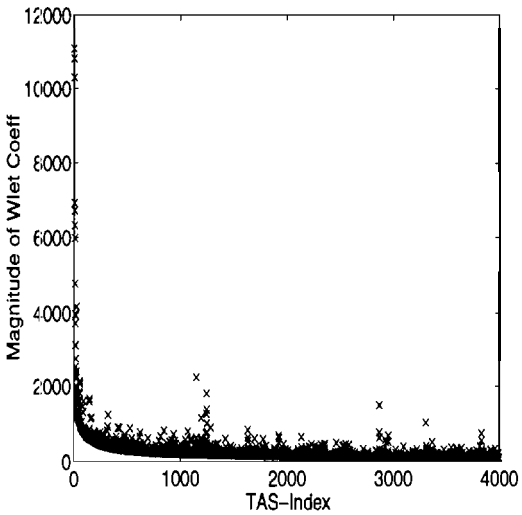
Fig. B.25. "Hotel" image with Haar Wavelet.



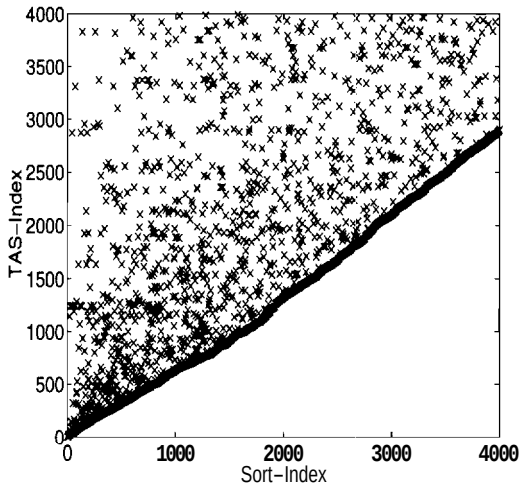
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

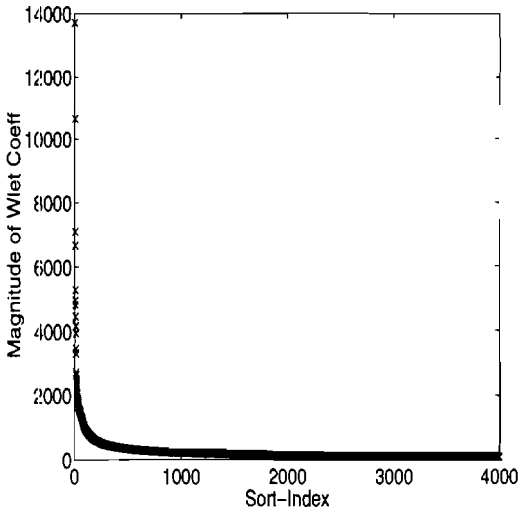


(c) Magnitudes vs TAS-Index

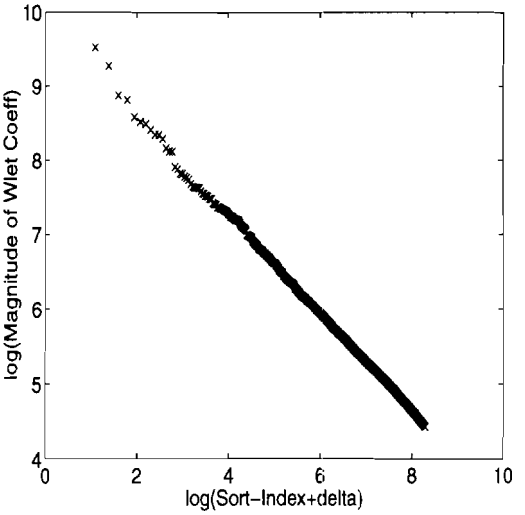


(d) TAS-Index vs Sort-Index

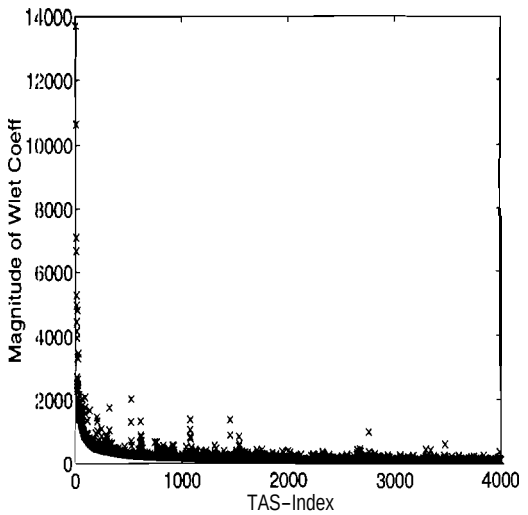
Fig. B.26. "Hotel" image with Daubechies D4 Wavelet.



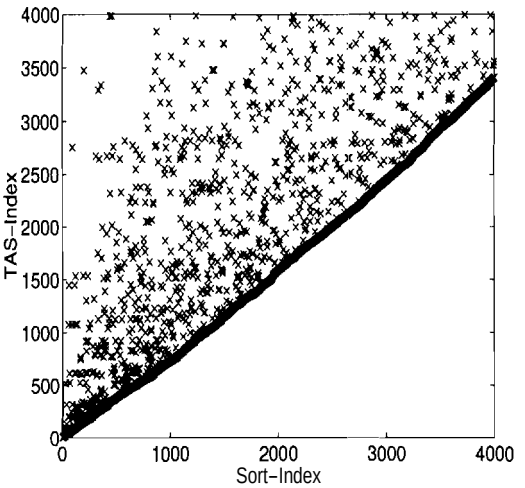
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

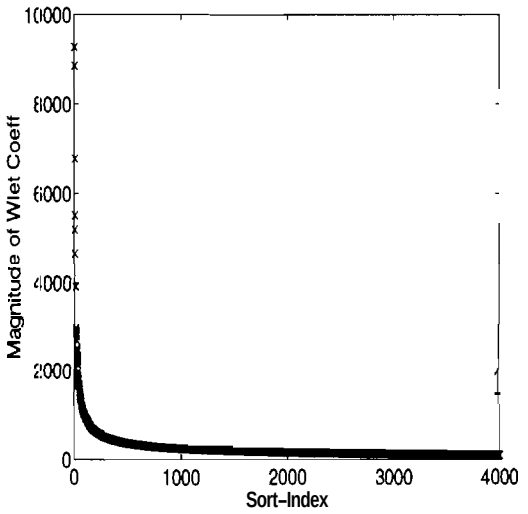


(c) Magnitudes vs TAS-Index

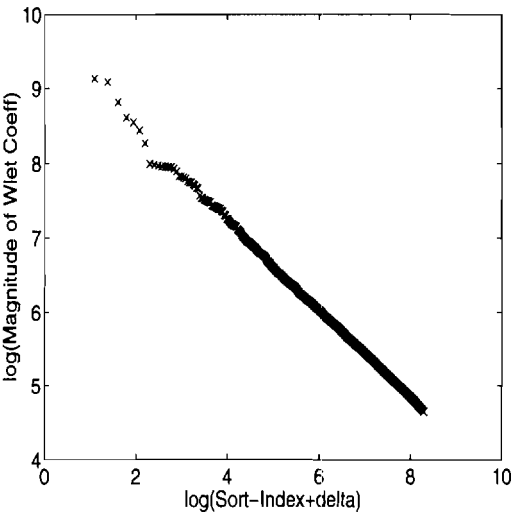


(d) TAS-Index vs Sort-Index

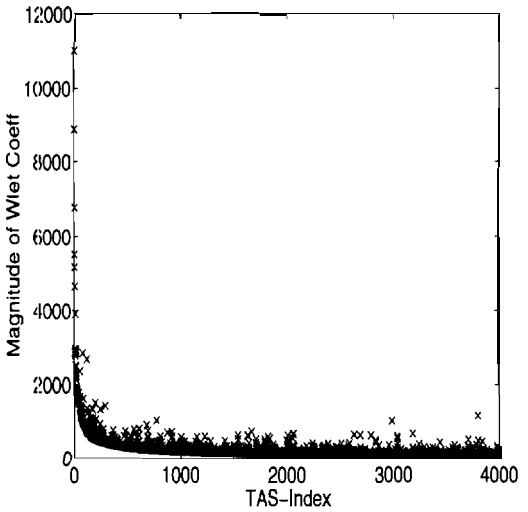
Fig. B.27. "Hotel" image with Spline-2 Wavelet.



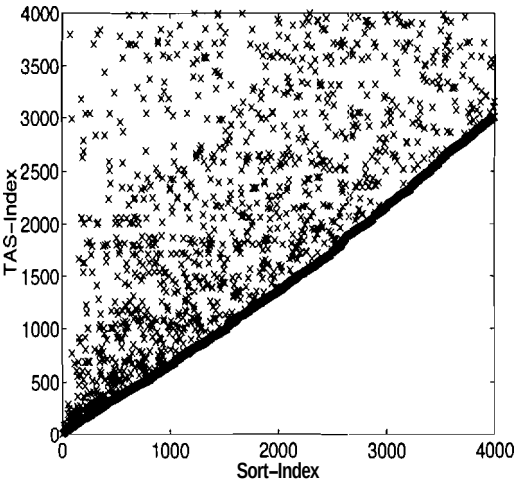
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

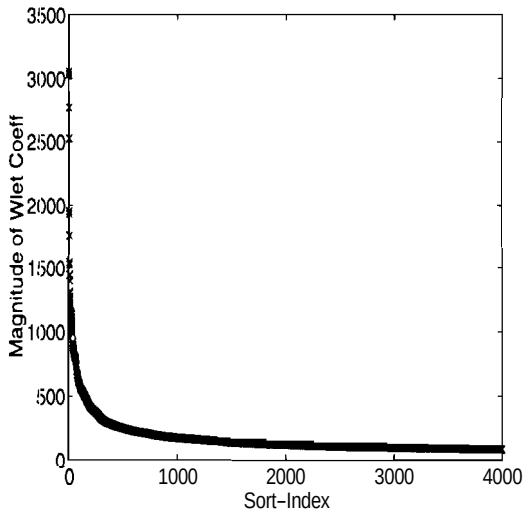


(c) Magnitudes vs TAS-Index

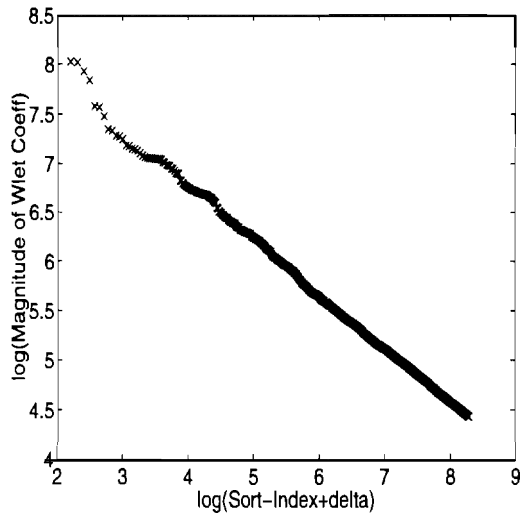


(d) TAS-Index vs Sort-Index

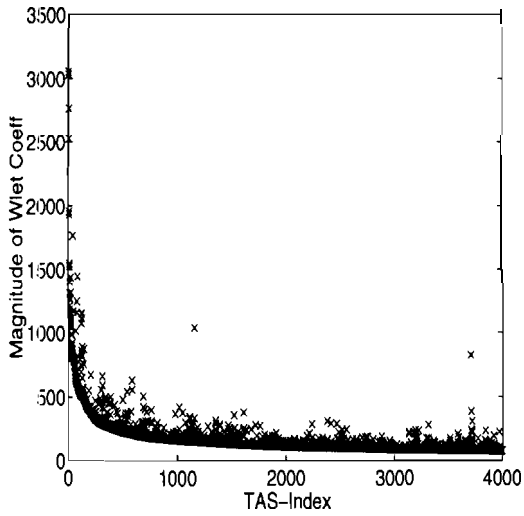
Fig. B.28. "Hotel" image with spline-variant Wavelet.



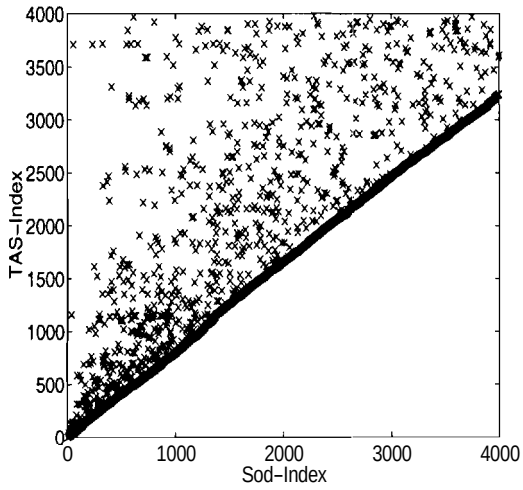
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

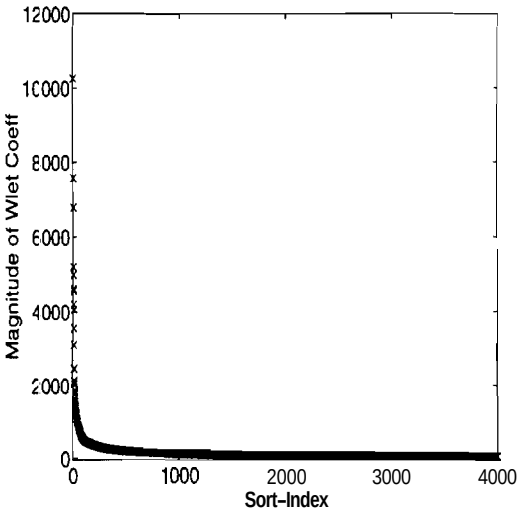


(c) Magnitudes vs TAS-Index

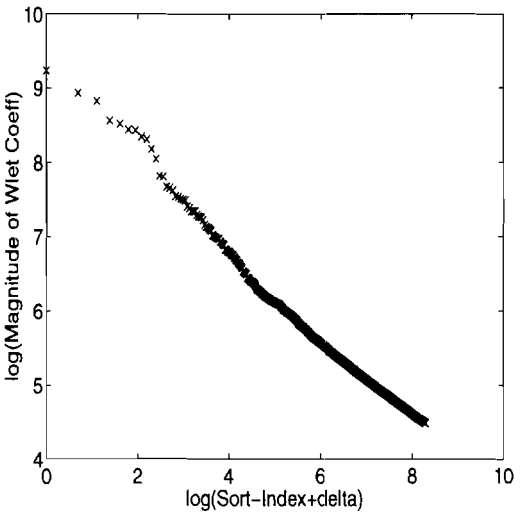


(d) TAS-Index vs Sort-Index

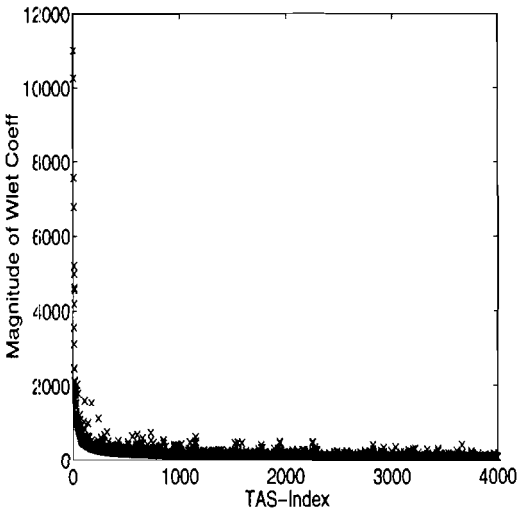
Fig. B.29. "Airport" image with Haar Wavelet.



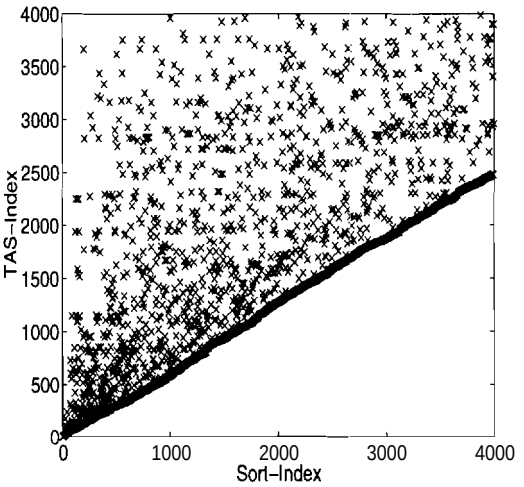
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

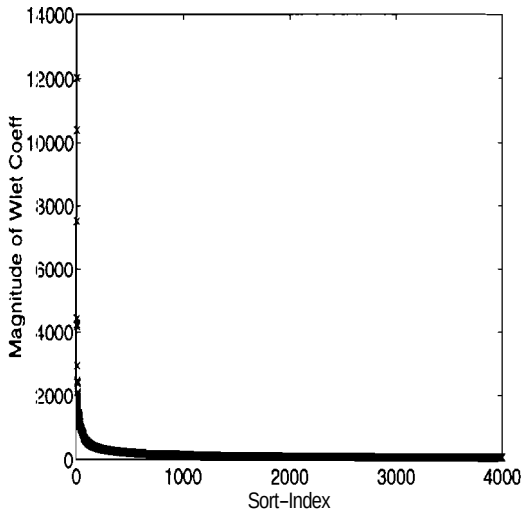


(c) Magnitudes vs TAS-Index

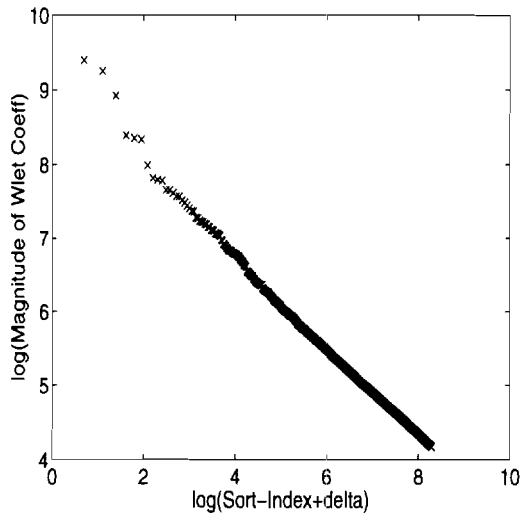


(d) TAS-Index vs Sort-Index

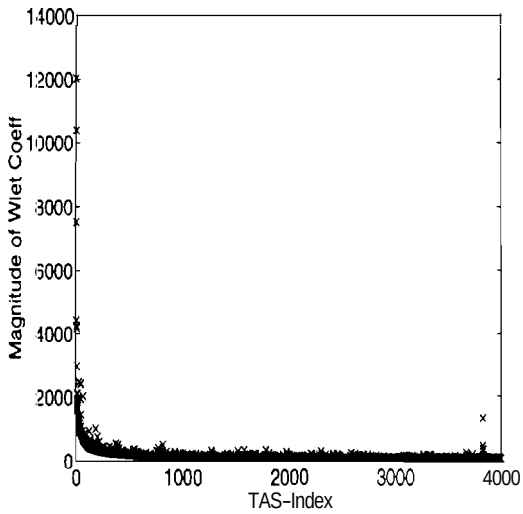
Fig. B.30. "Airport" image with Daubechies D4 Wavelet.



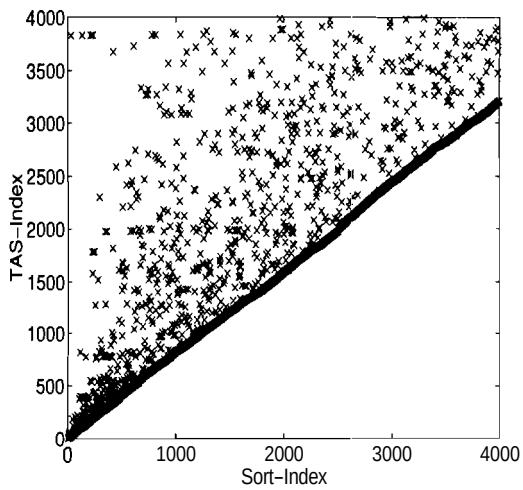
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

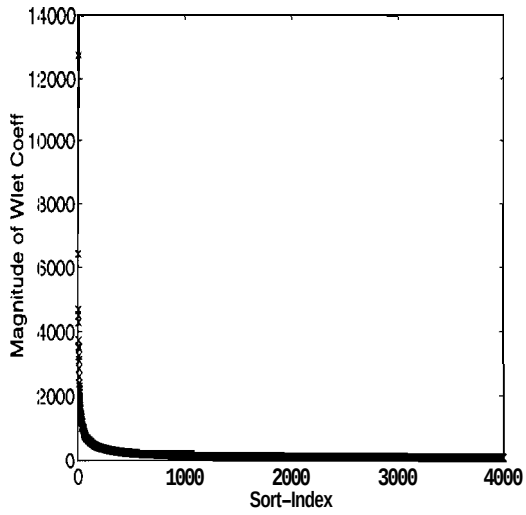


(c) Magnitudes vs TAS-Index

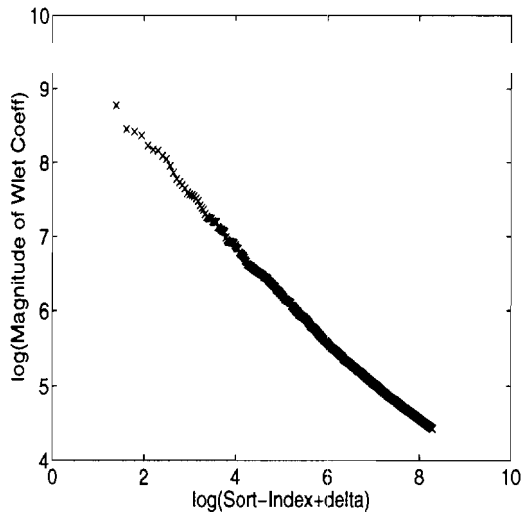


(d) TAS-Index vs Sort-Index

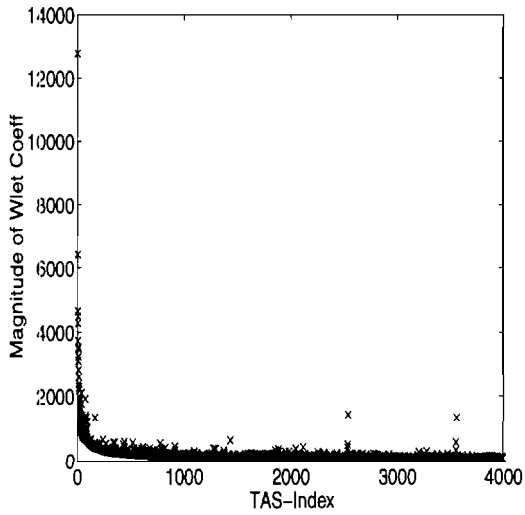
Fig. B.31. "Airport" image with Spline-2 Wavelet.



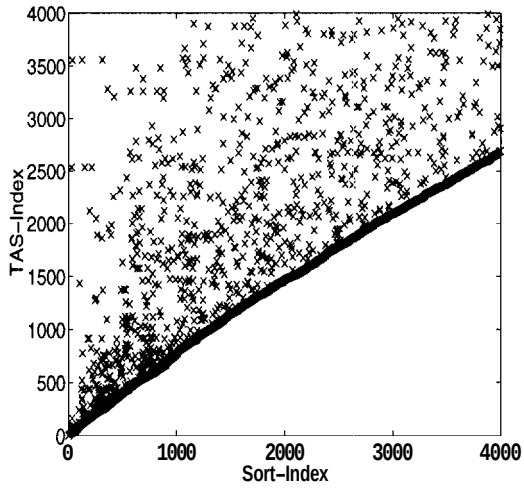
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

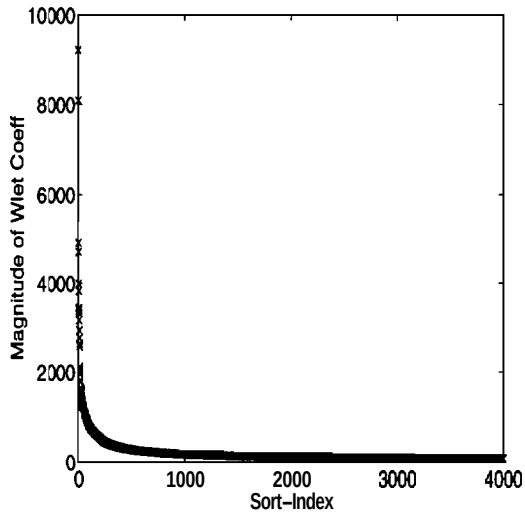


(c) Magnitudes vs TAS-Index

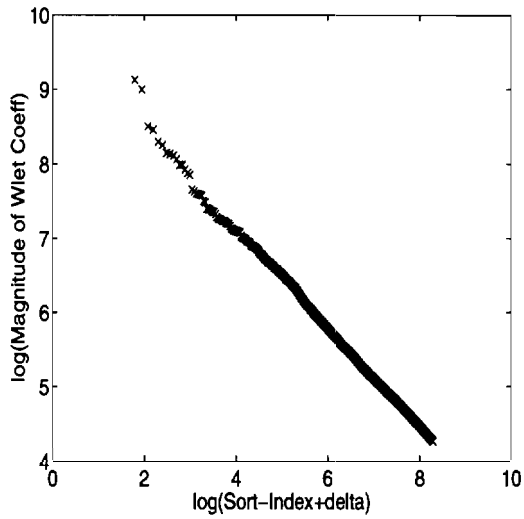


(d) TAS-Index vs Sort-Index

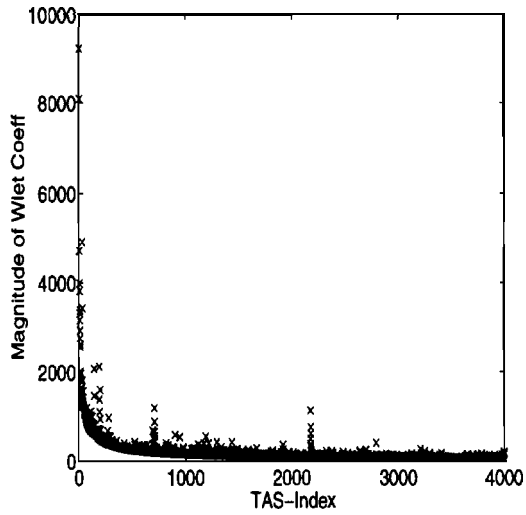
Fig. B.32. "Airport" image with spline-variant Wavelet.



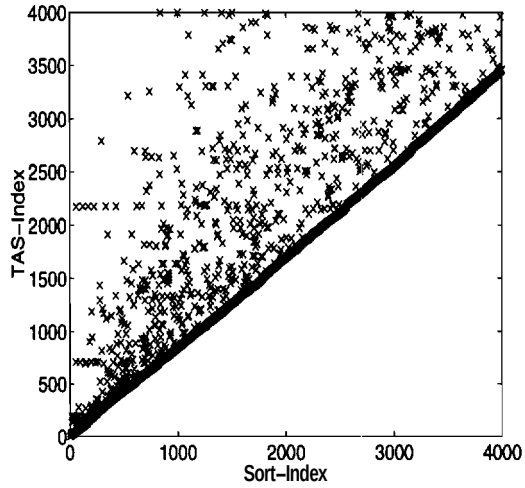
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

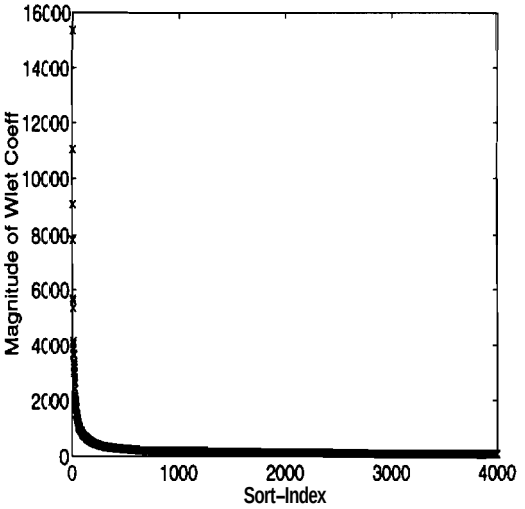


(c) Magnitudes vs TAS-Index

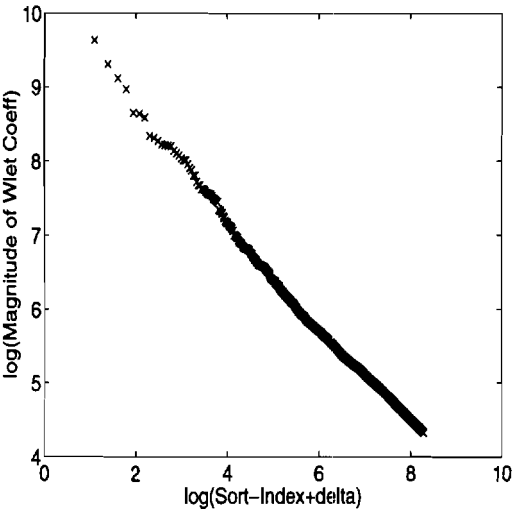


(d) TAS-Index vs Sort-Index

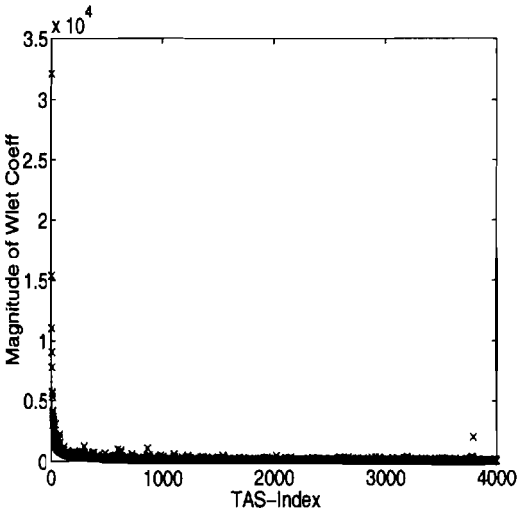
Fig. **B.33.** "Mustang" image with Haar Wavelet.



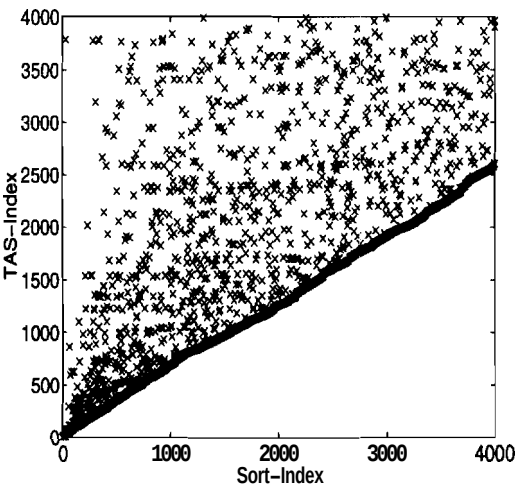
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

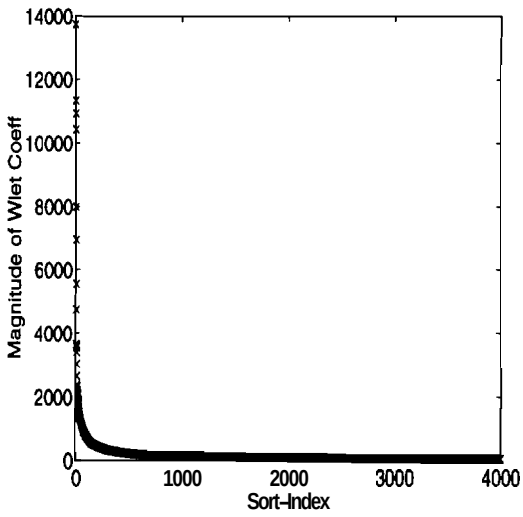


(c) Magnitudes vs TAS-Index

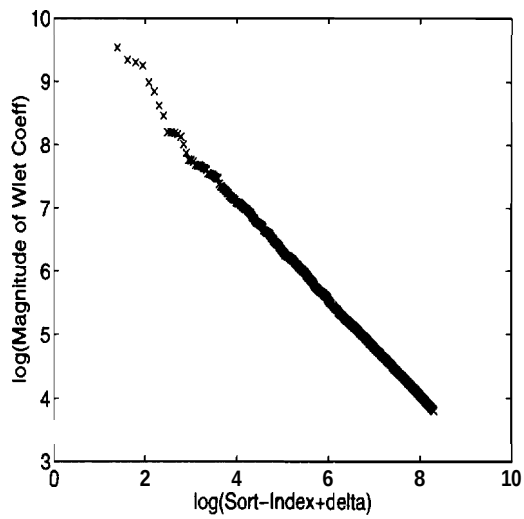


(d) TAS-Index vs Sort-Index

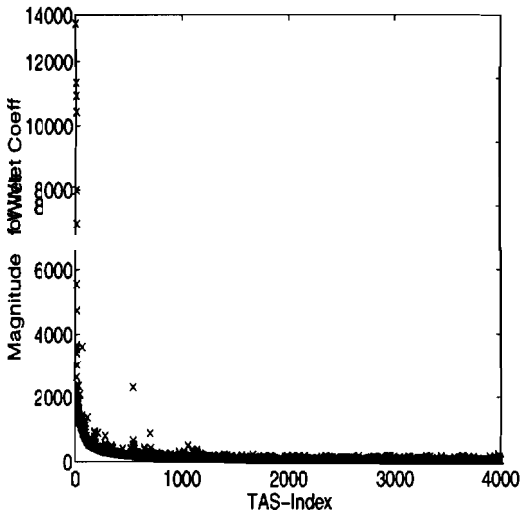
Fig. B.34. "Mustang" image with Daubechies D4 Wavelet.



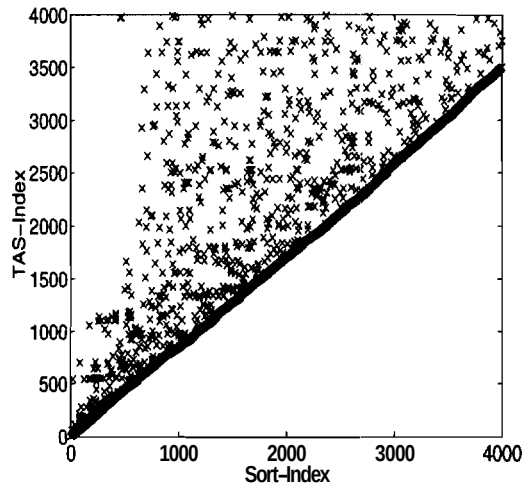
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

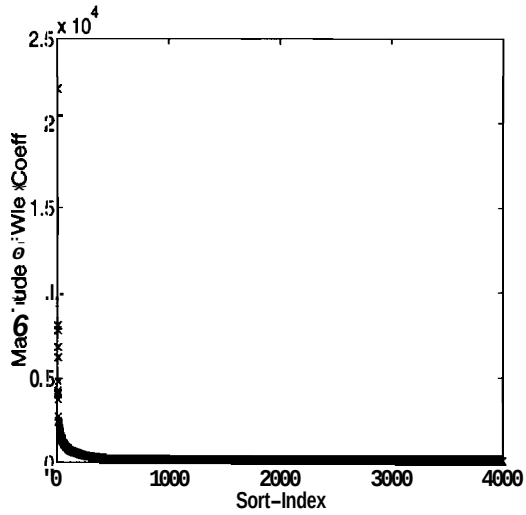


(c) Magnitudes vs TAS-Index

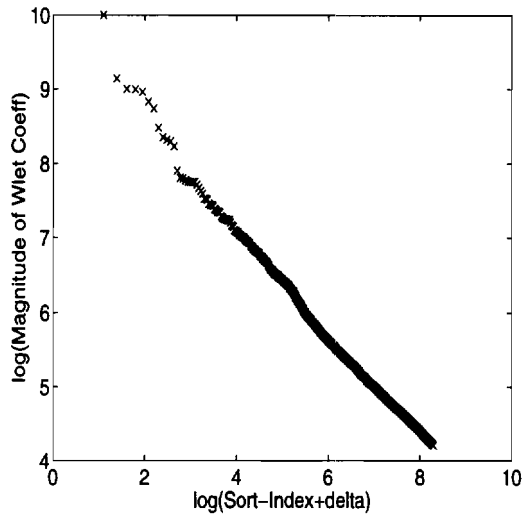


(d) TAS-Index vs Sort-Index

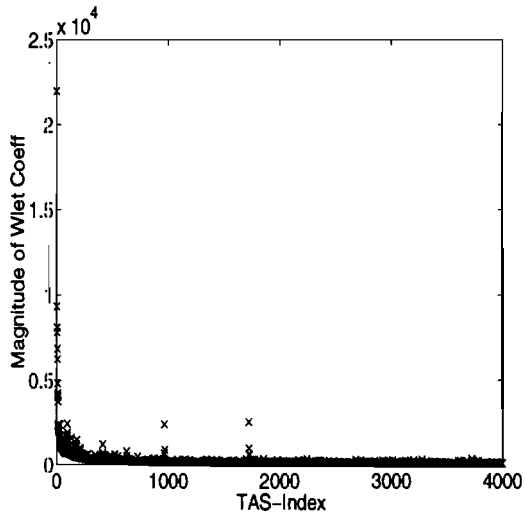
Fig. B.35. "Mustang" image with Spline-2 Wavelet.



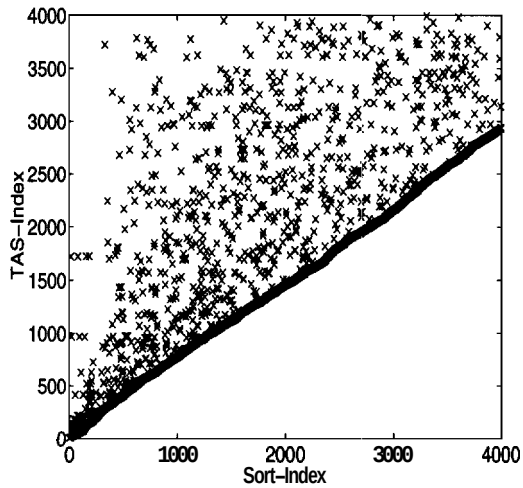
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

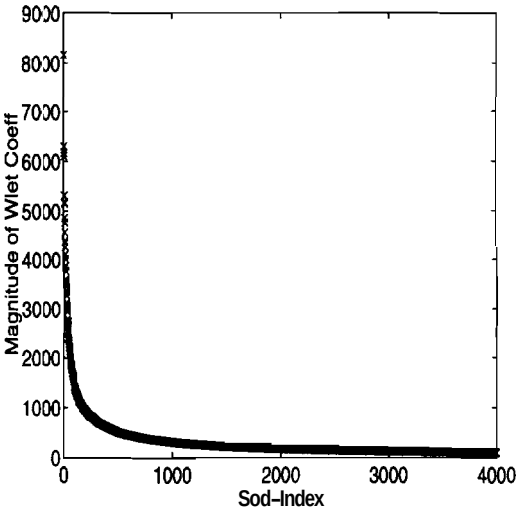


(c) Magnitudes vs TAS-Index

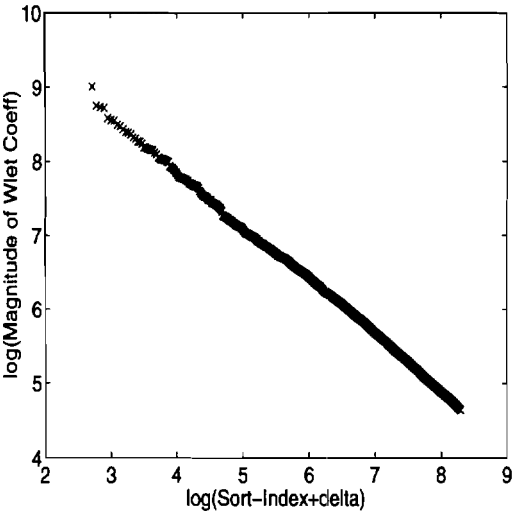


(d) TAS-Index vs Sort-Index

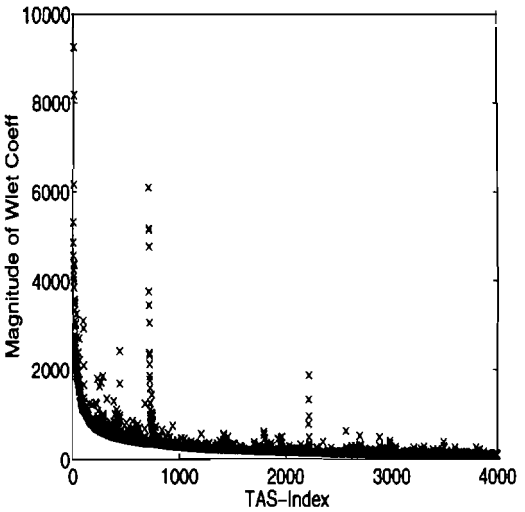
Fig. B.36. "Mustang" image with spline-variant Wavelet.



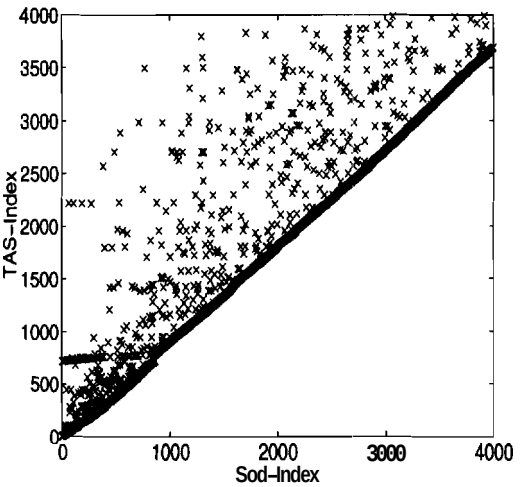
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

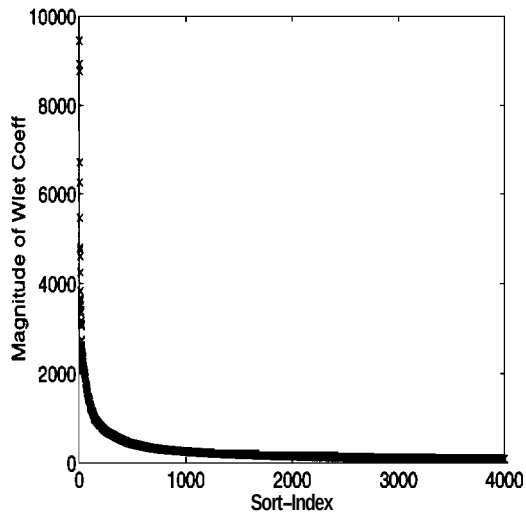


(c) Magnitudes vs TAS-Index

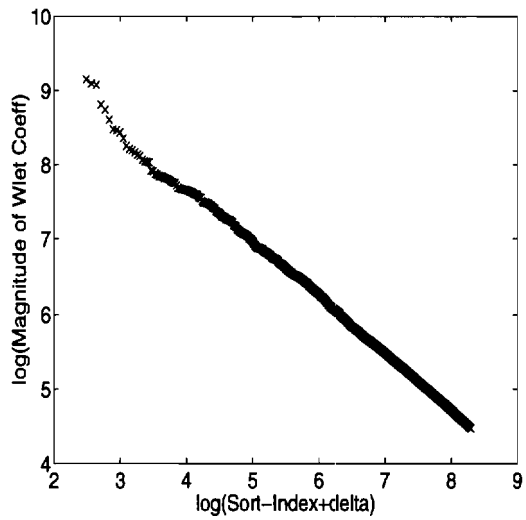


(d) TAS-Index vs Sort-.Index

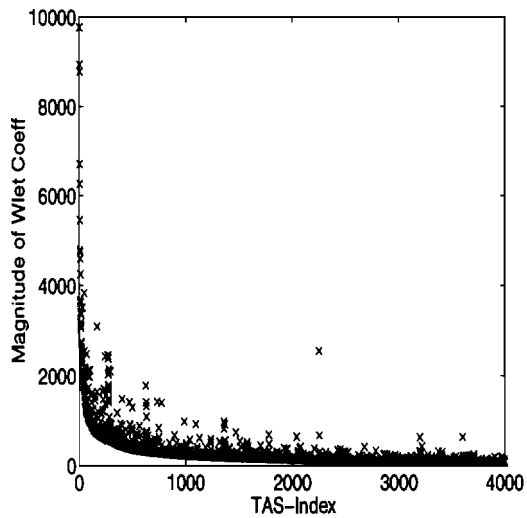
Fig. **B.37.** "Peppers" image with Haar Wavelet.



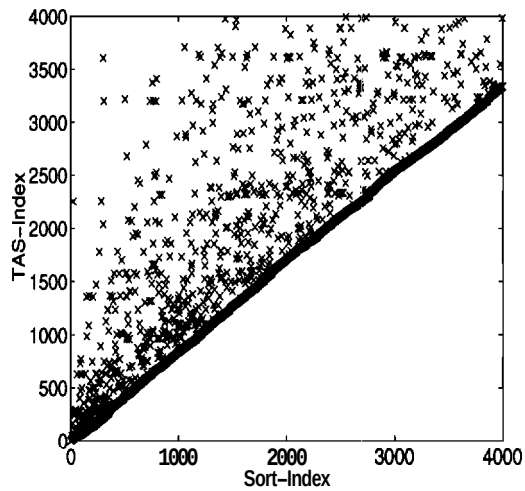
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

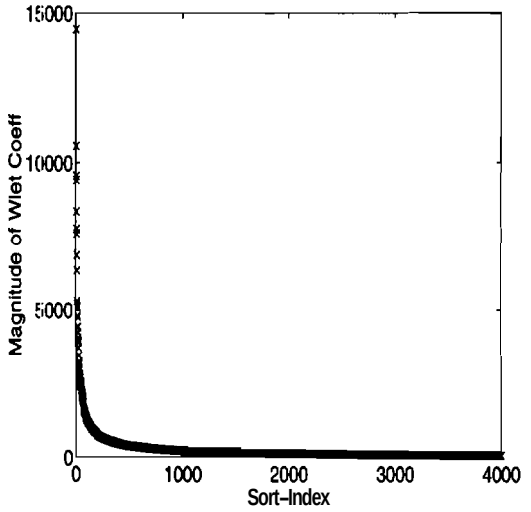


(c) Magnitudes vs TAS-Index

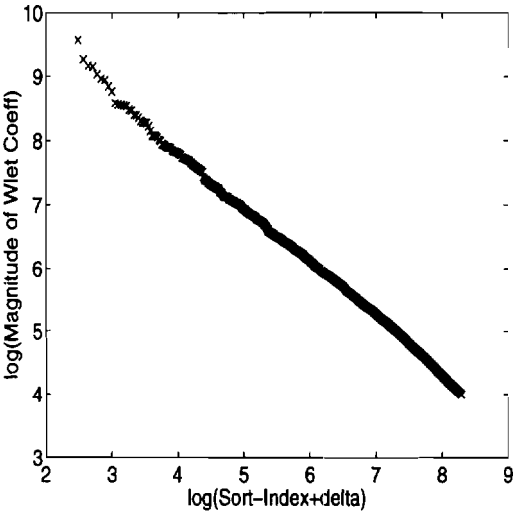


(d) TAS-Index vs Sort-Index

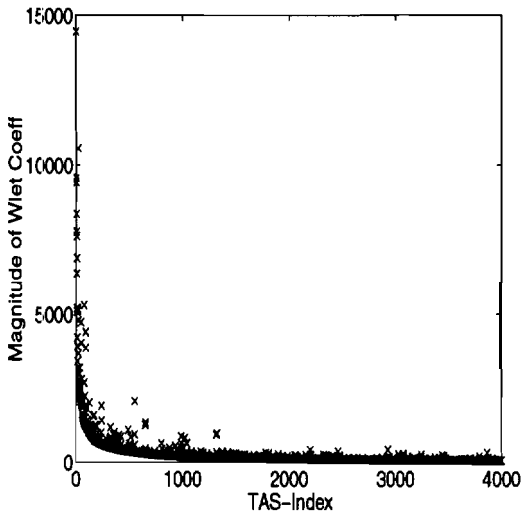
Fig. B.38. "Peppers" image with Daubechies D4 Wavelet.



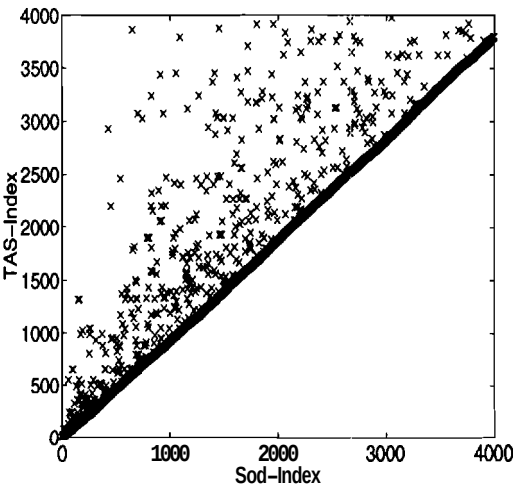
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

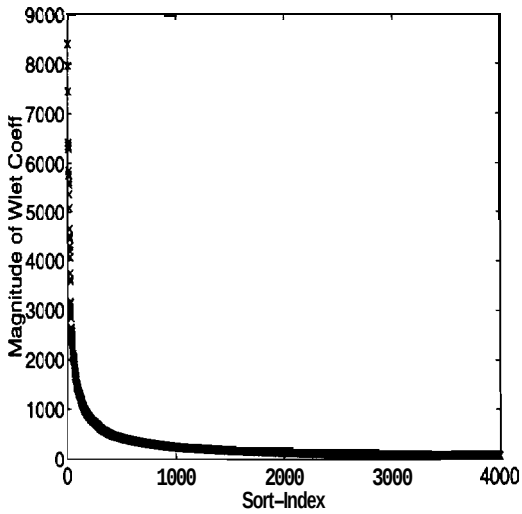


(c) Magnitudes vs TAS-Index

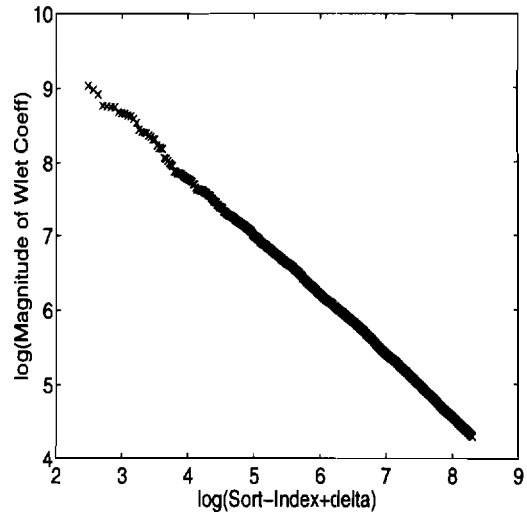


(d) TAS-Index vs Sort-.Index

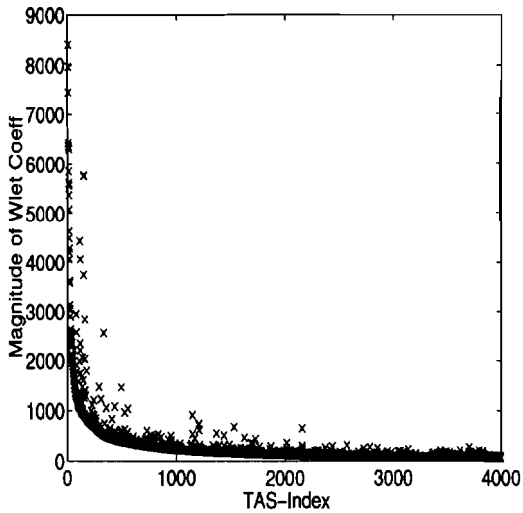
Fig. **B.39.** "Peppers" image with Spline-2 Wavelet.



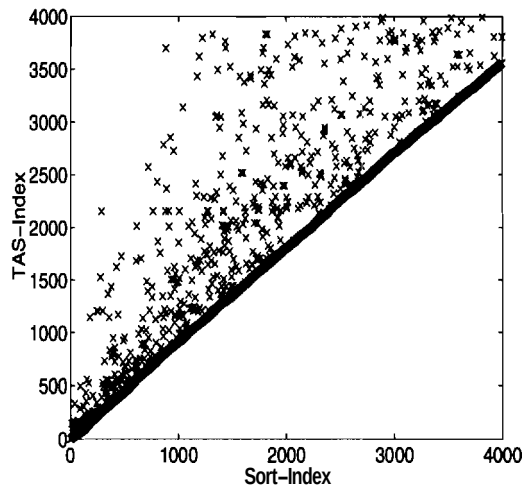
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

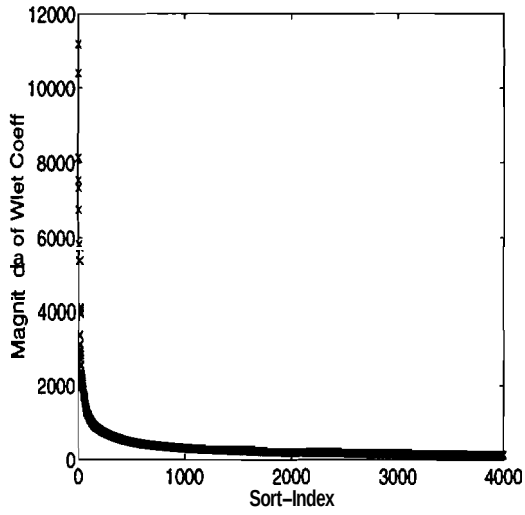


(c) Magnitudes vs TAS-Index

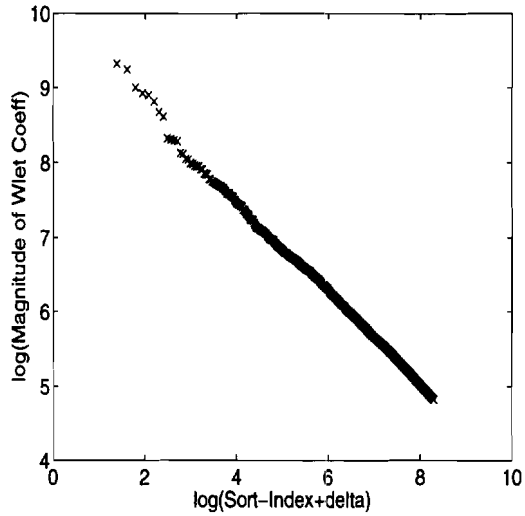


(d) TAS-Index vs Sort-Index

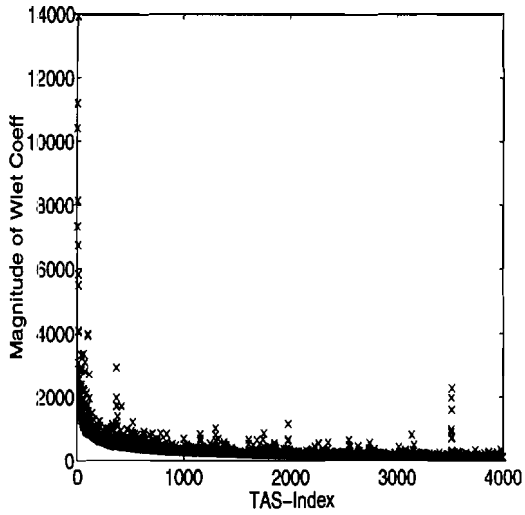
Fig. B.40. "Peppers" image with spline-variant Wavelet;.



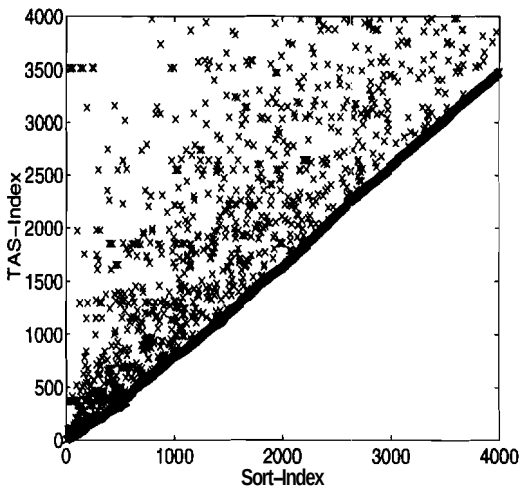
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

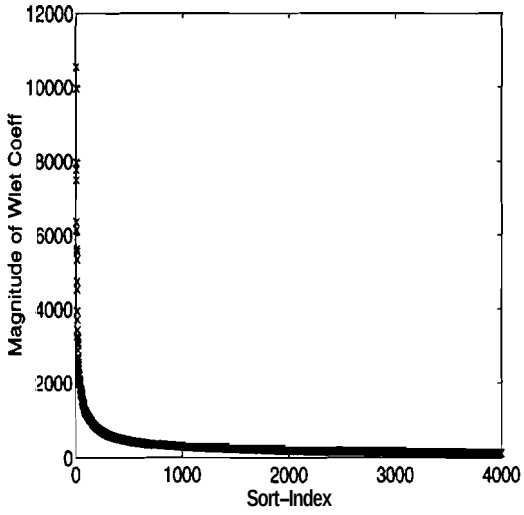


(c) Magnitudes vs TAS-Index

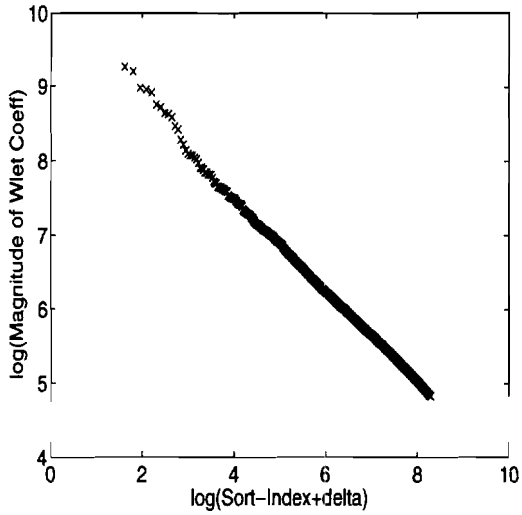


(d) TAS-Index vs Sort-Index

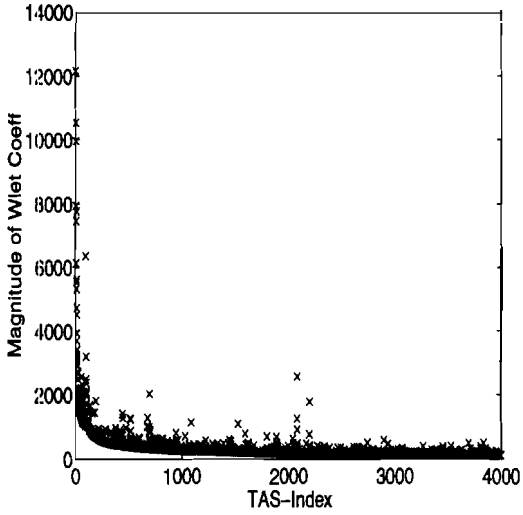
Fig. B.41. "Sailboat" image with Haar Wavelet.



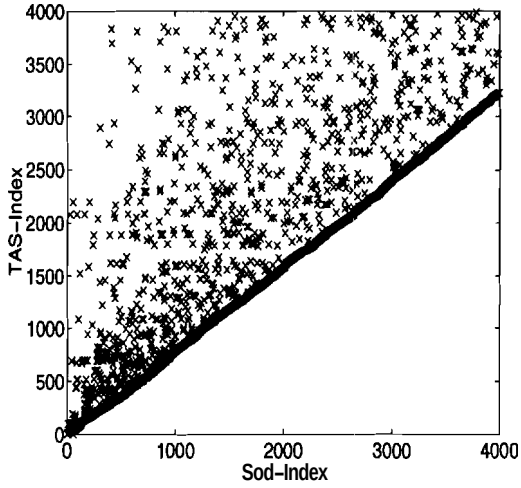
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

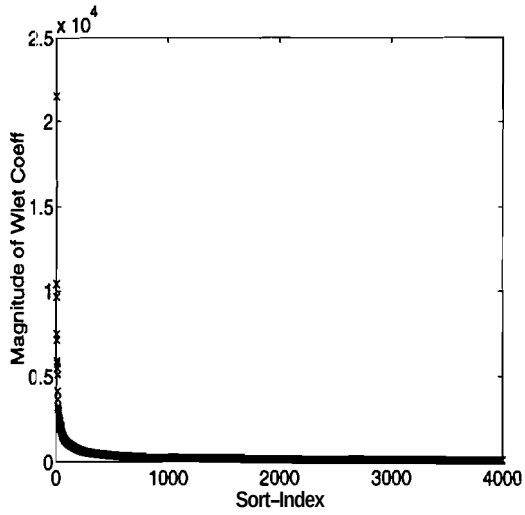


(c) Magnitudes vs TAS-Index

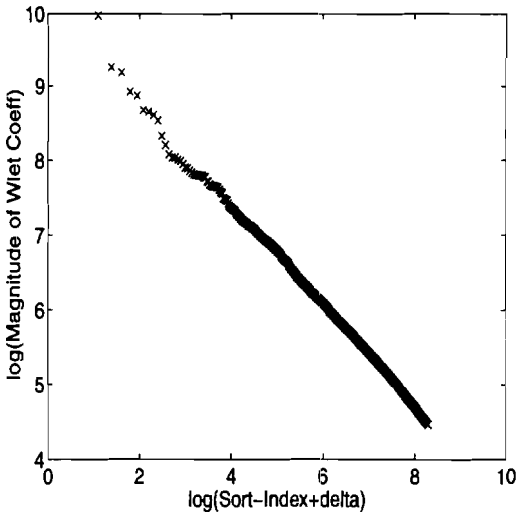


(d) TAS-Index vs Sort-Index

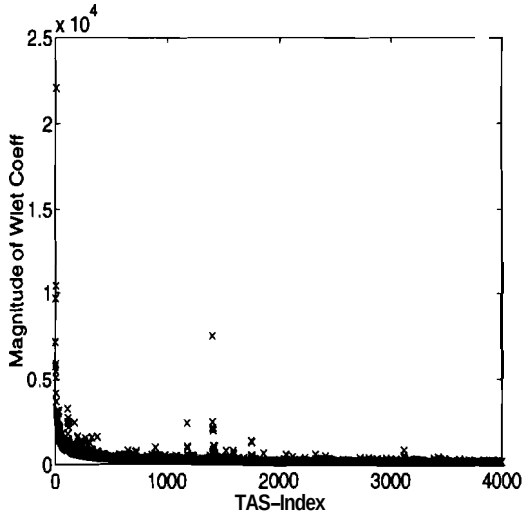
Fig. B.42. "Sailboat" image with Daubechies D4 Wavelet.



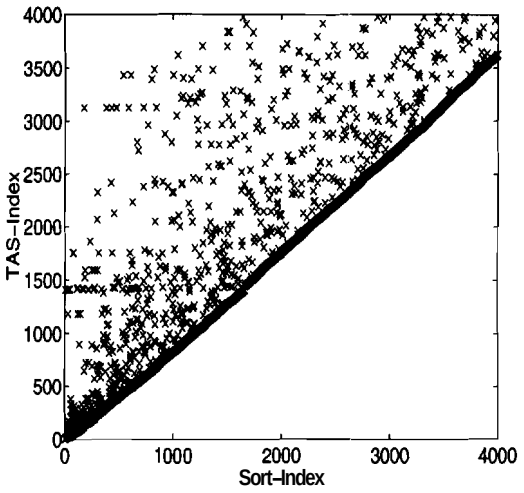
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

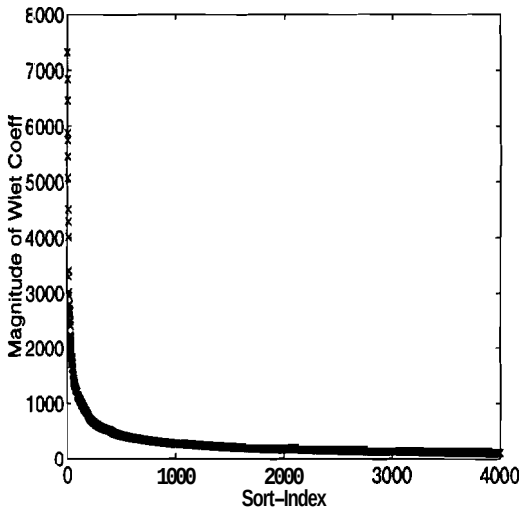


(c) Magnitudes vs TAS-Index

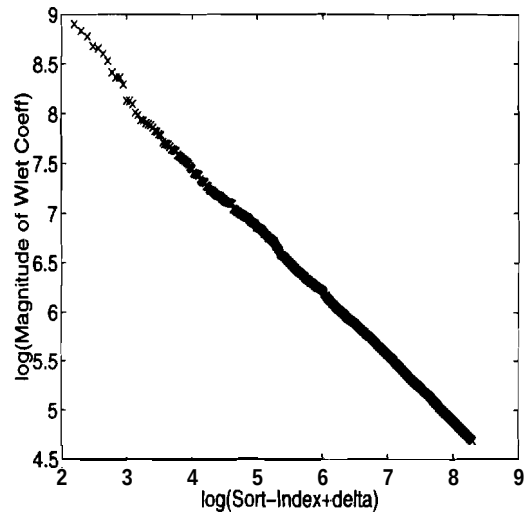


(d) TAS-Index vs Sort-Index

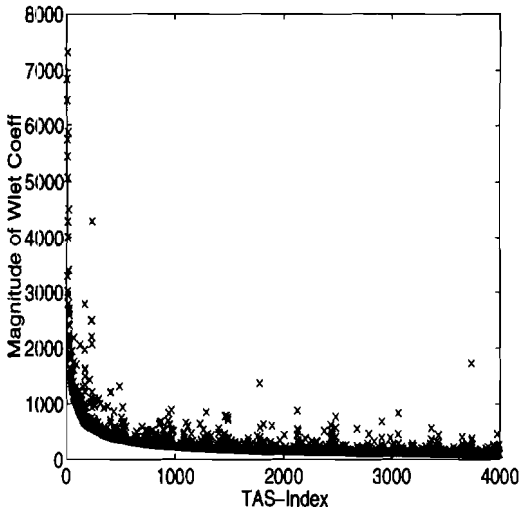
Fig. **B.43.** "Sailboat" image with Spline-2 Wavelet.



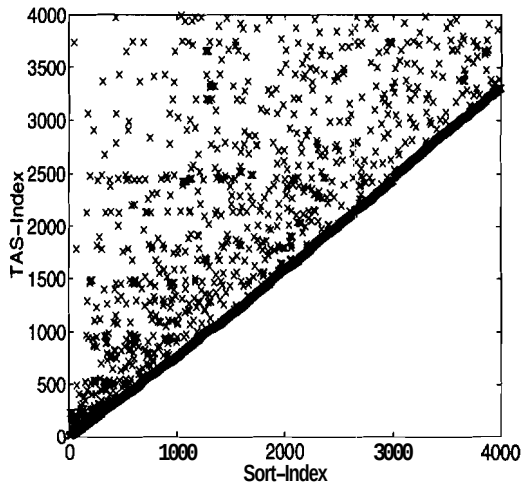
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

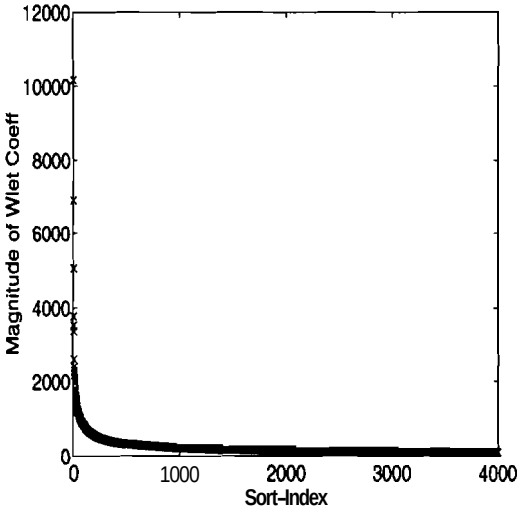


(c) Magnitudes vs TAS-Index

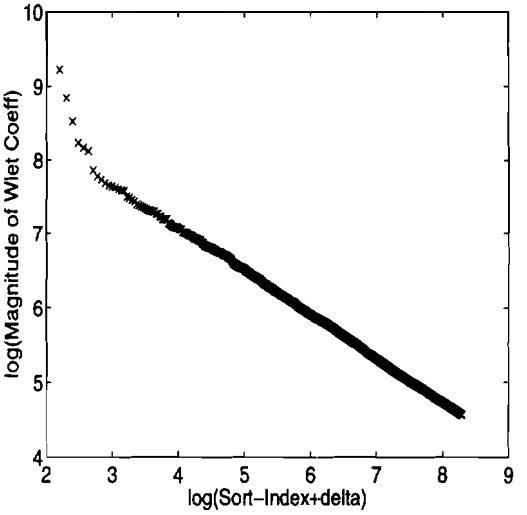


(d) TAS-Index vs Sort-Index

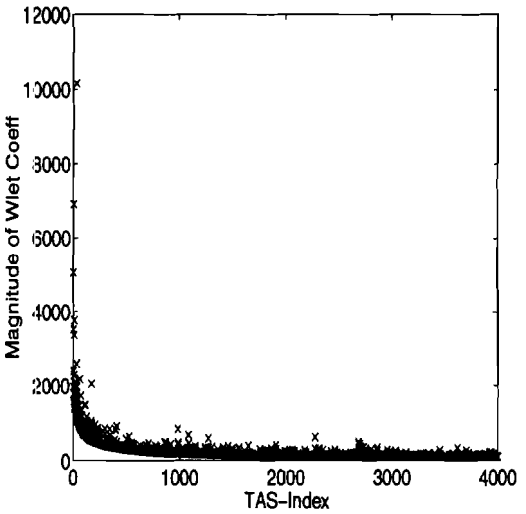
Fig. B.44. "Sailboat" image with spline-variant Wavelet.



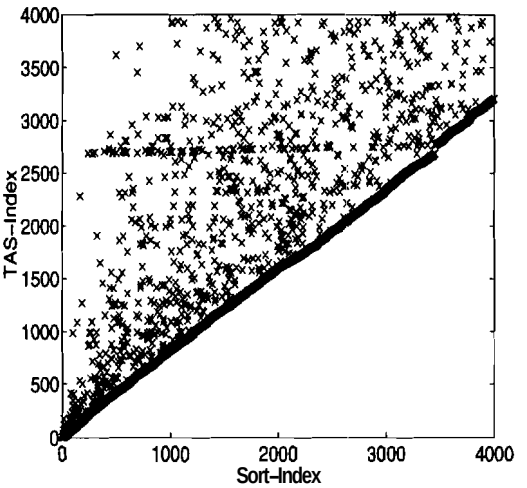
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

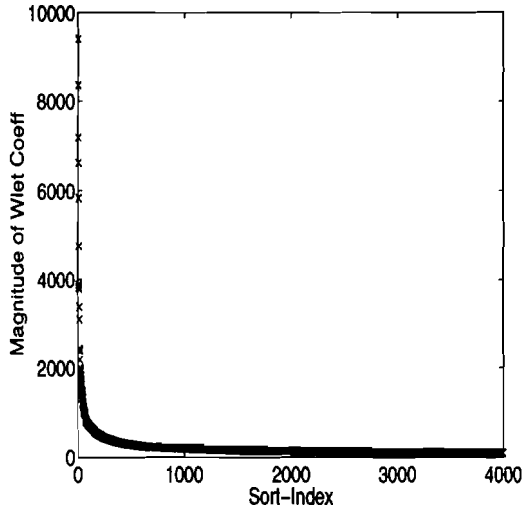


(c) Magnitudes vs TAS-Index

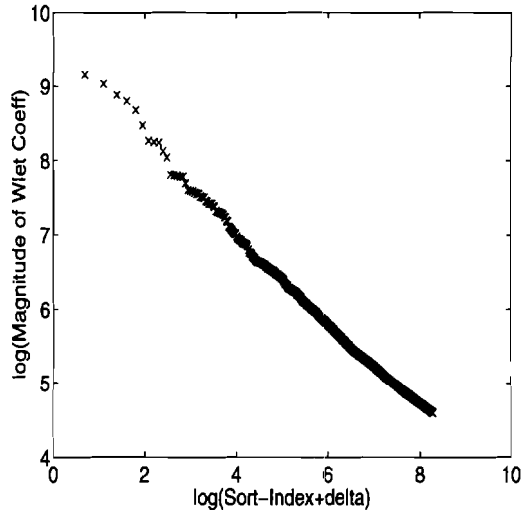


(d) TAS-Index vs Sort-Index

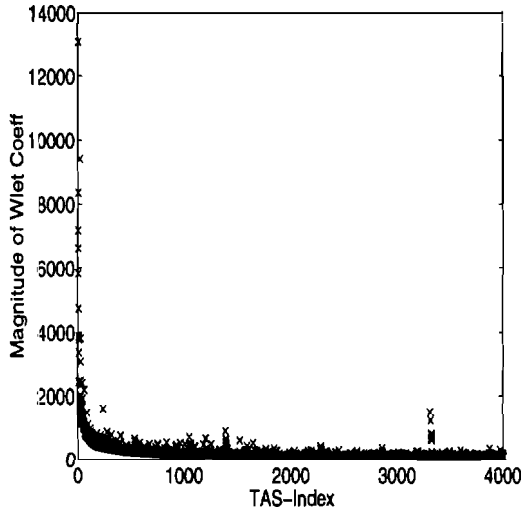
Fig. B.45. "Stream" image with Haar Wavelet.



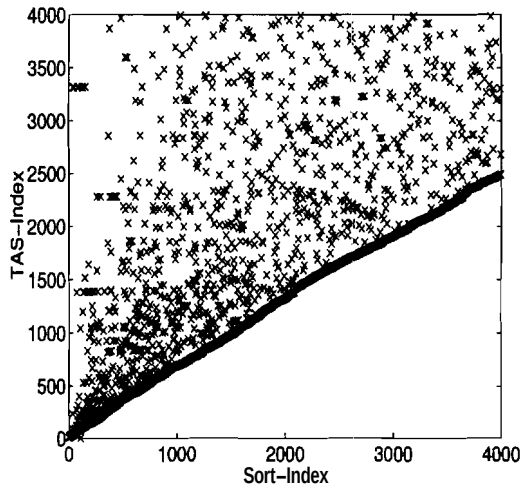
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

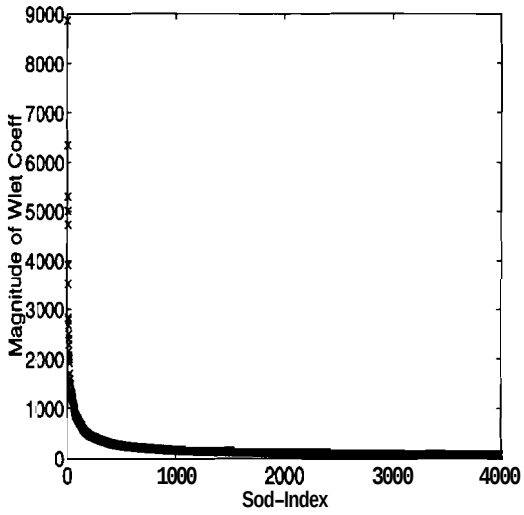


(c) Magnitudes vs TAS-Index

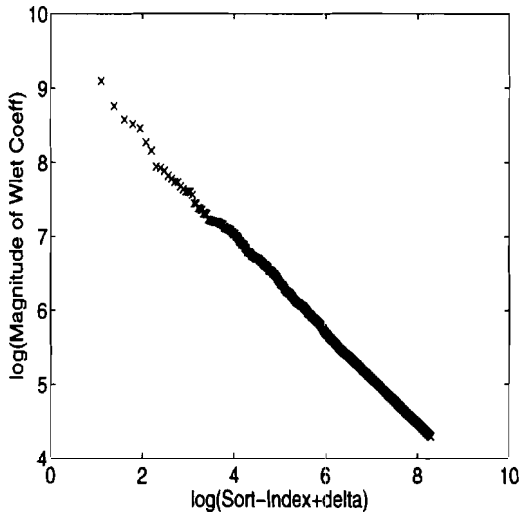


(d) TAS-Index vs Sort-Index

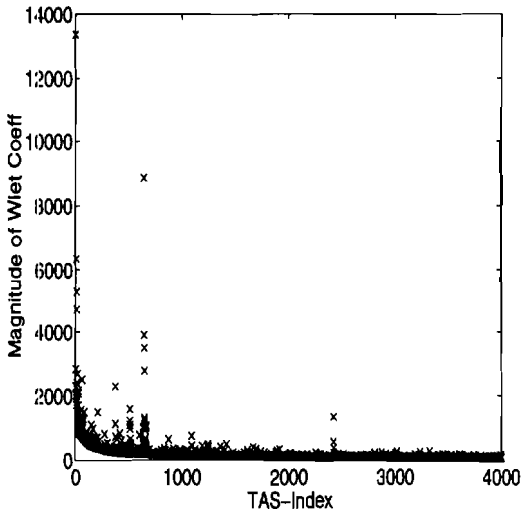
Fig. B.46. "Stream" image with Daubechies D4 Wavelet.



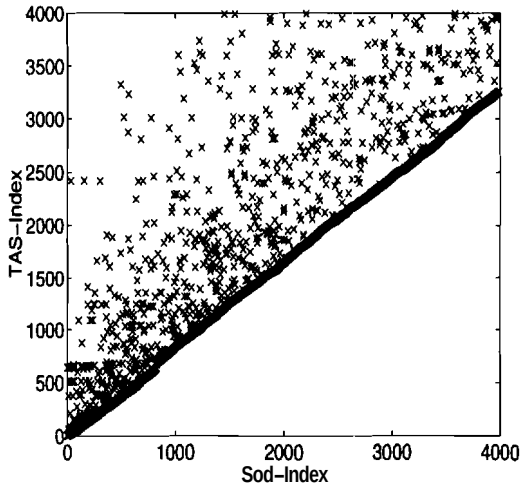
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

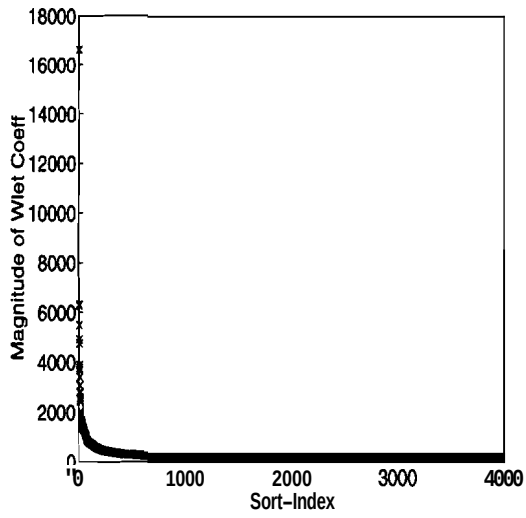


(c) Magnitudes vs TAS-Index

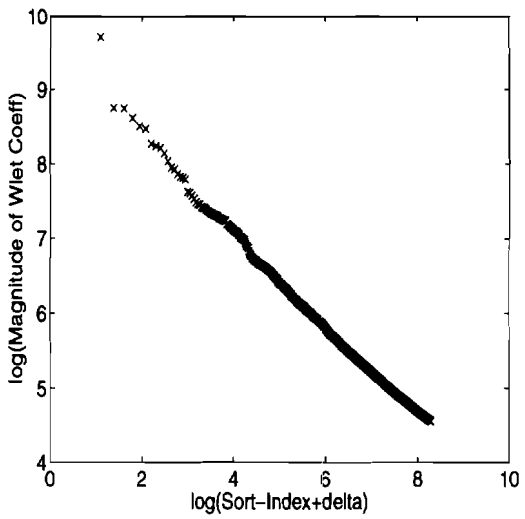


(d) TAS-Index vs Sort-Index

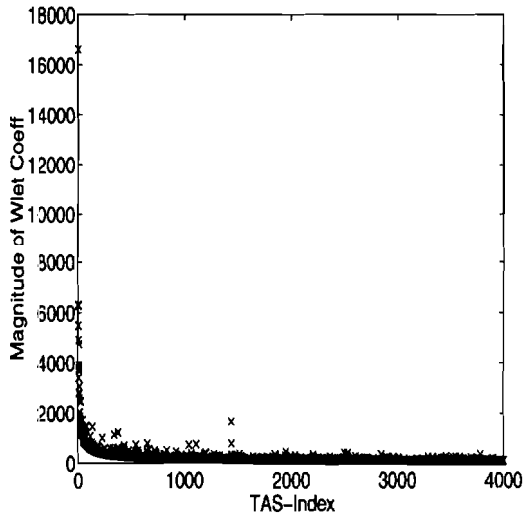
Fig. B.47. "Stream" image with Spline-2 Wavelet.



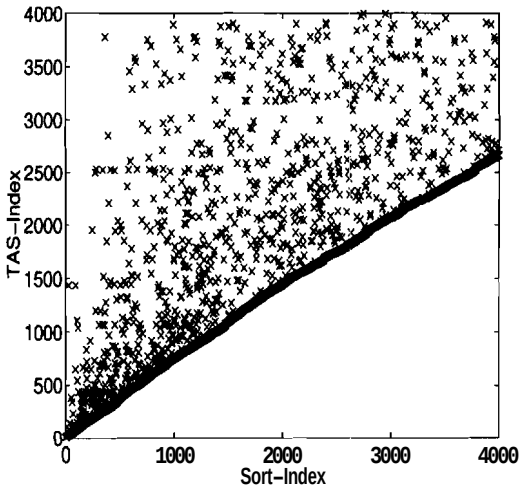
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

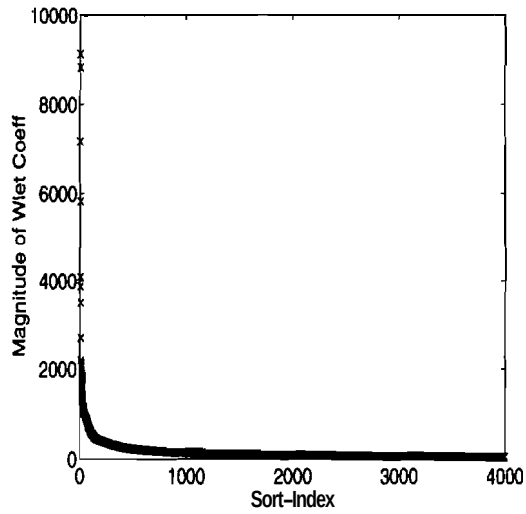


(c) Magnitudes vs TAS-Index

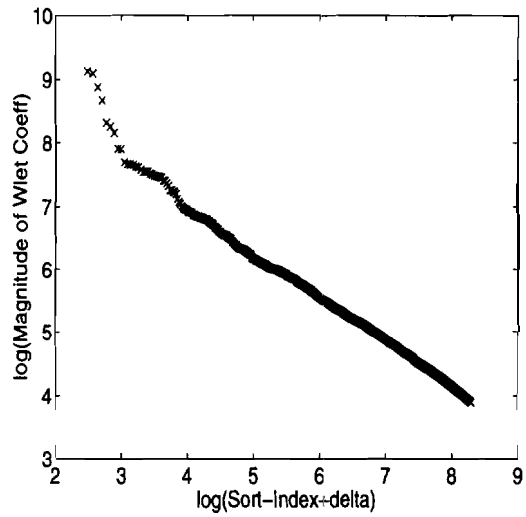


(d) TAS-Index vs Sort-Index

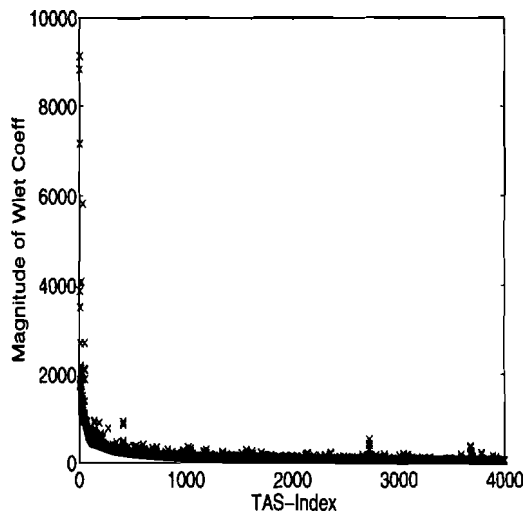
Fig. B.48. "Stream" image with spline-variant Wavelet.



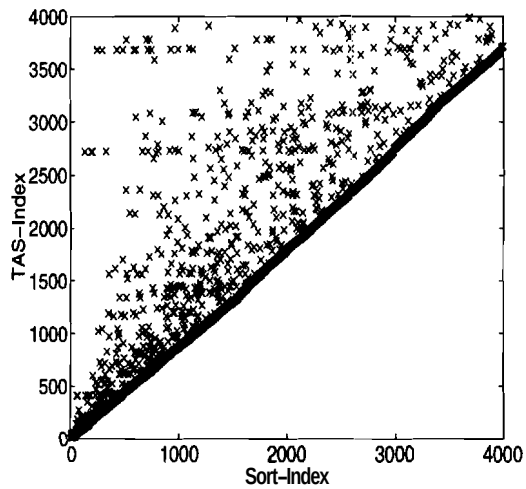
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

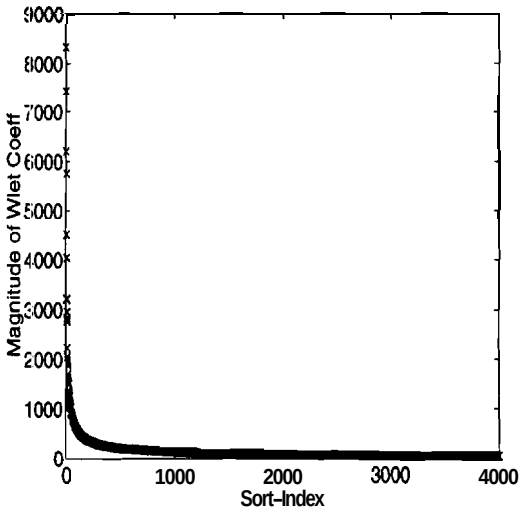


(c) Magnitudes vs TAS-Index

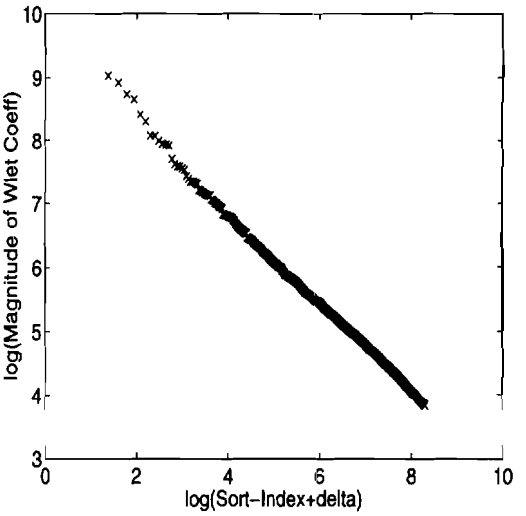


(d) TAS-Index vs Sort-Index

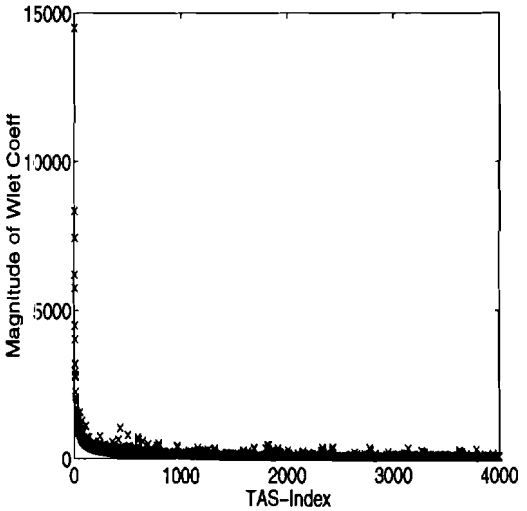
Fig. B.49. "widget" image with Haar Wavelet.



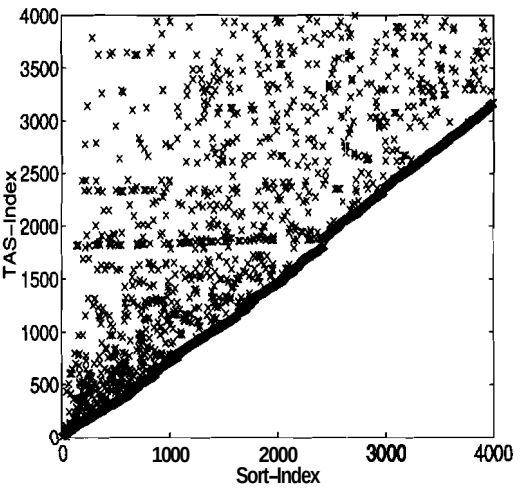
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

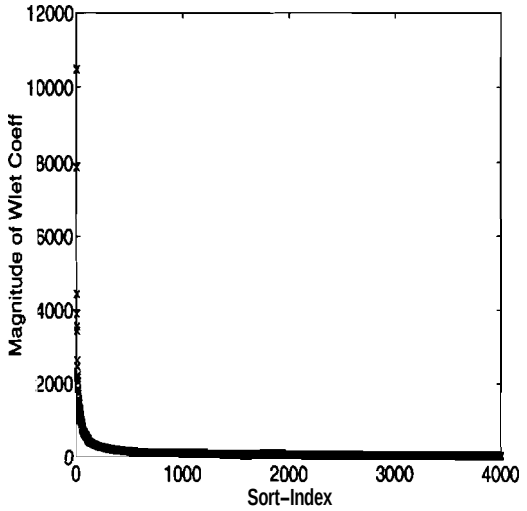


(c) Magnitudes vs TAS-Index

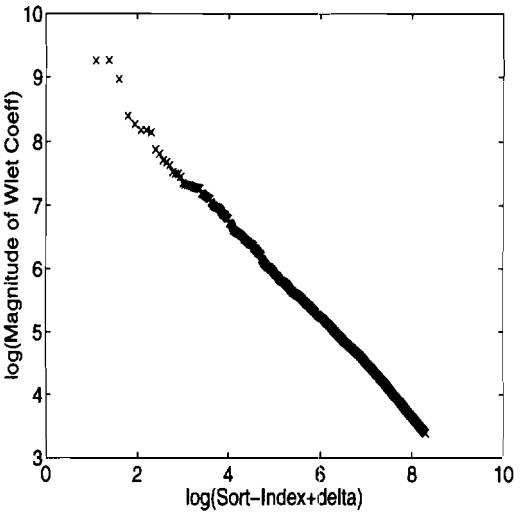


(d) TAS-Index vs Sort.-Index

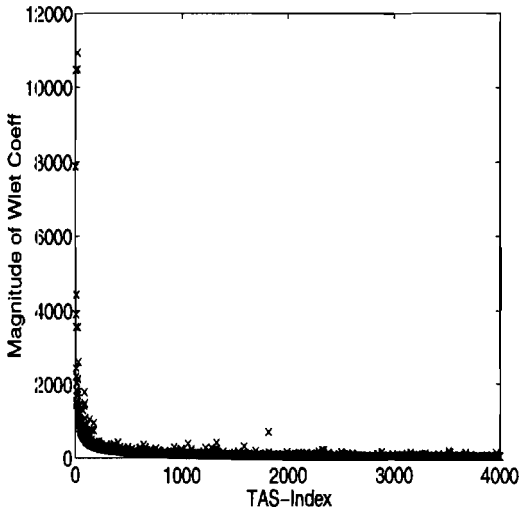
Fig. B.50. "widget" image with Daubechies D4 Wavelet.



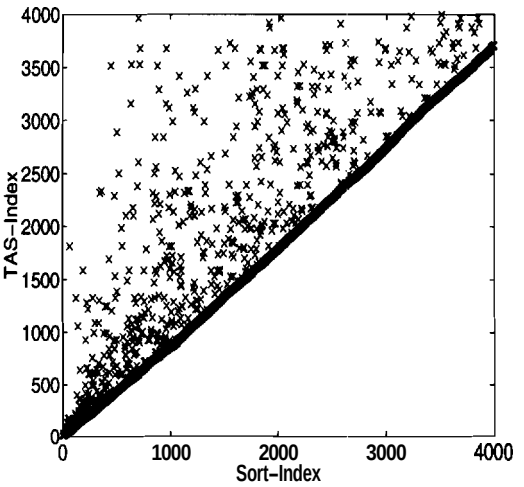
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

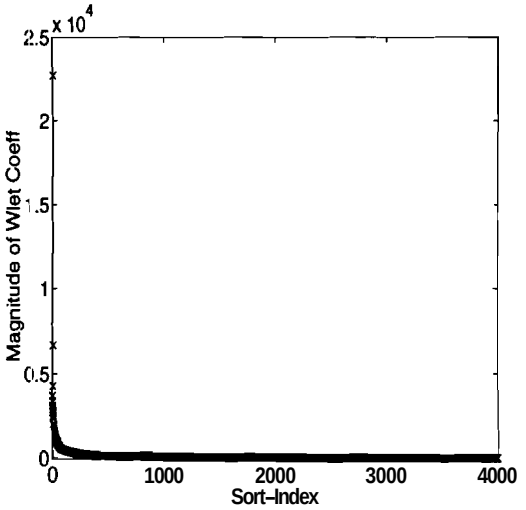


(c) Magnitudes vs TAS-Index

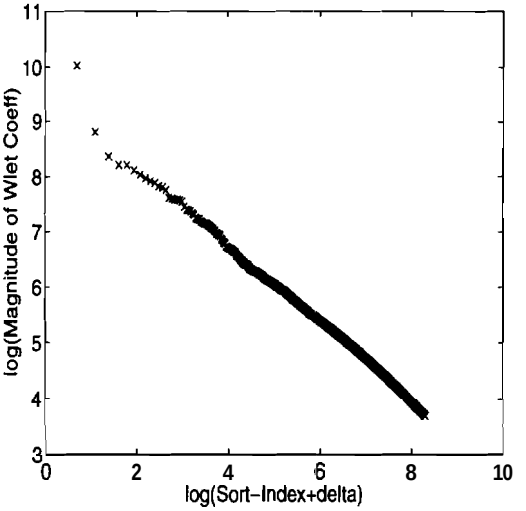


(d) TAS-Index vs Sort.-Index

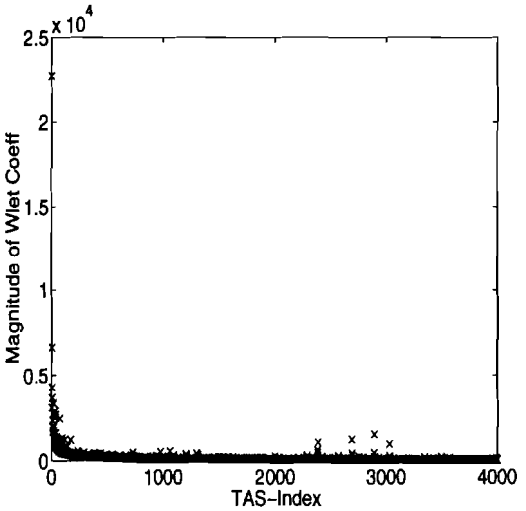
Fig. B.51. "widget" image with Spline-2 Wavelet.



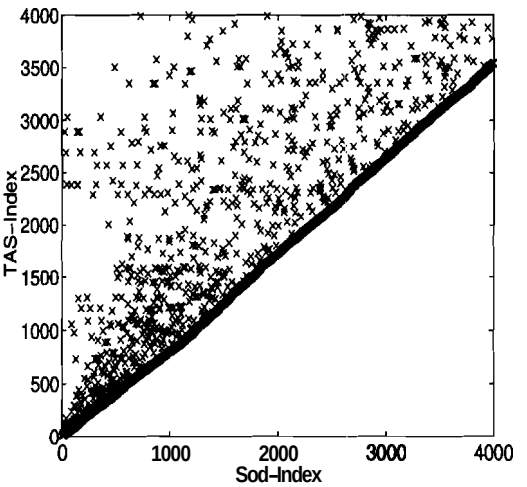
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

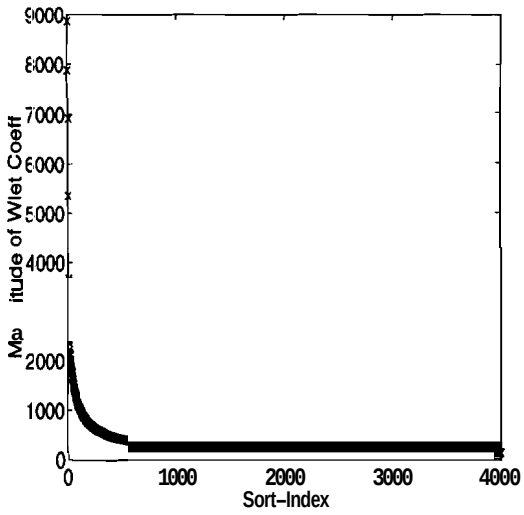


(c) Magnitudes vs TAS-Index

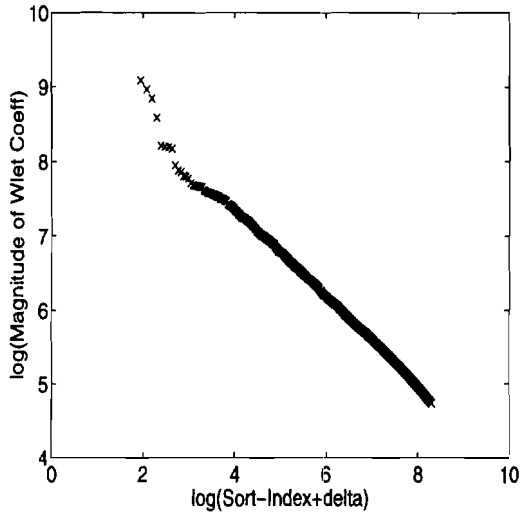


(d) TAS-Index vs Sort-Index

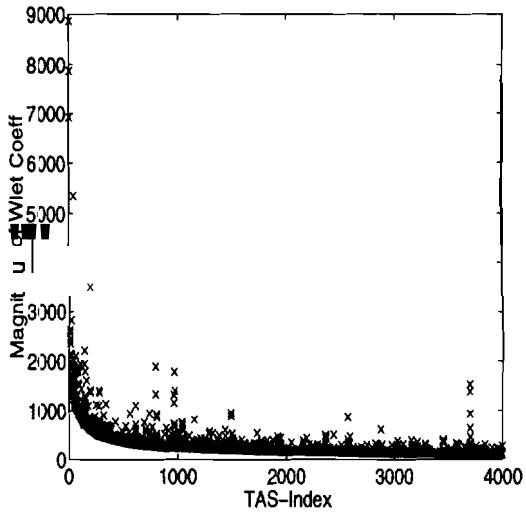
Fig. B.52. "widget" image with spline-variant Wavelet.



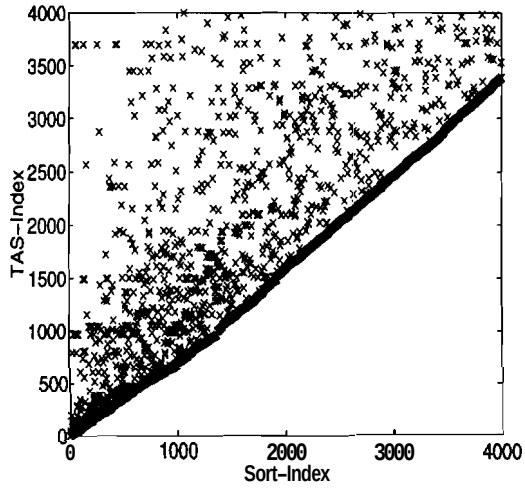
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

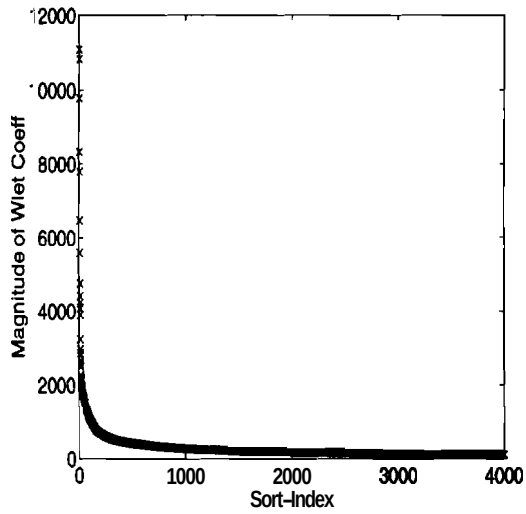


(c) Magnitudes vs TAS-Index

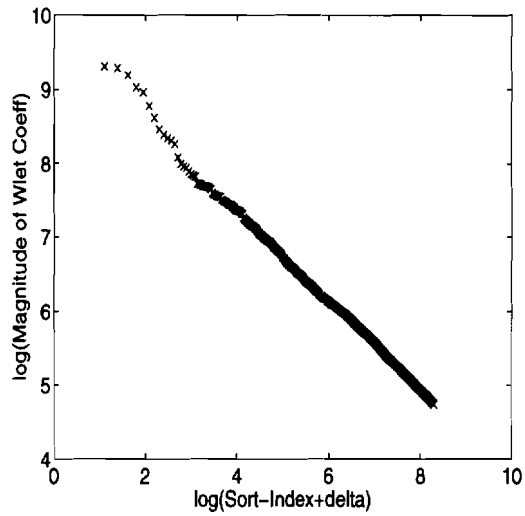


(d) TAS-Index vs Sort-Index

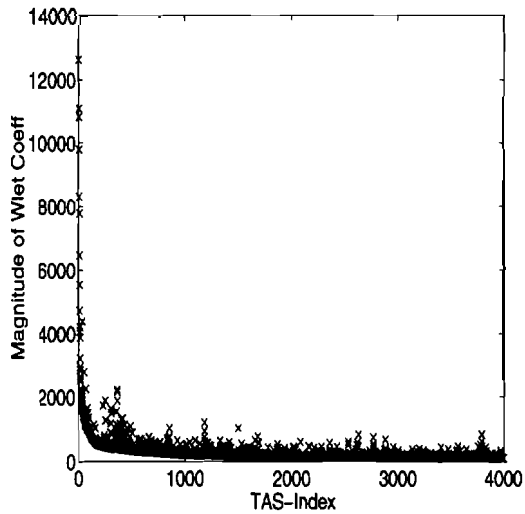
Fig. B.53. "bank" image with Haar Wavelet.



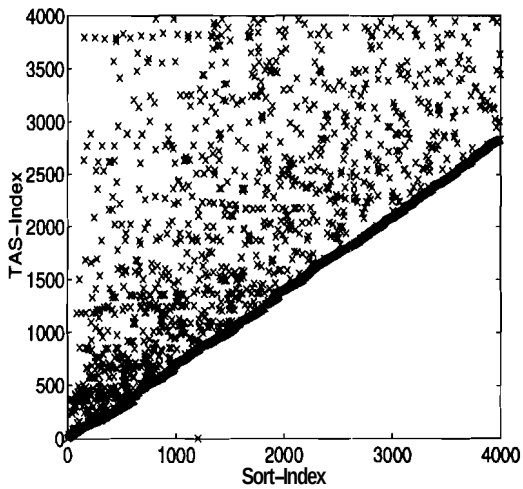
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

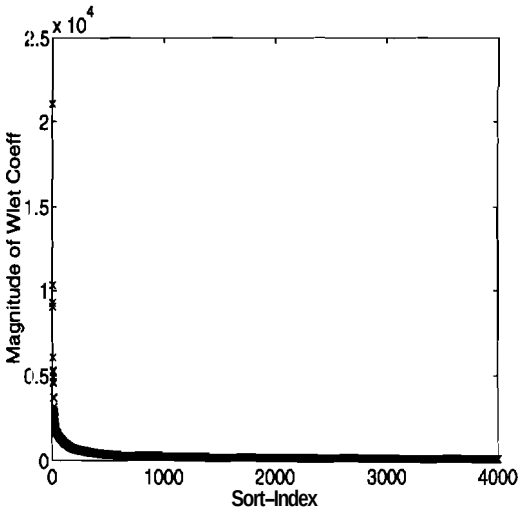


(c) Magnitudes vs TAS-Index

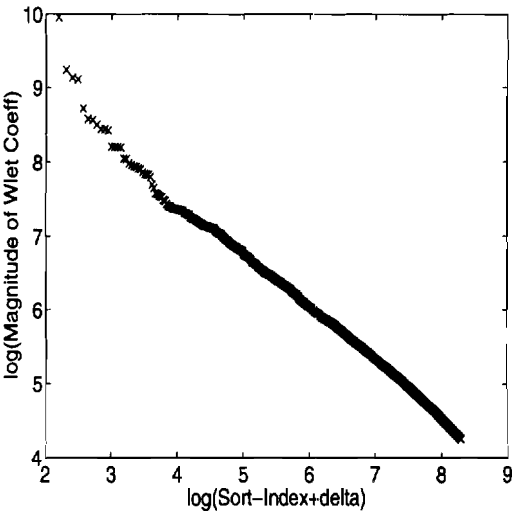


(d) TAS-Index vs Sort-Index

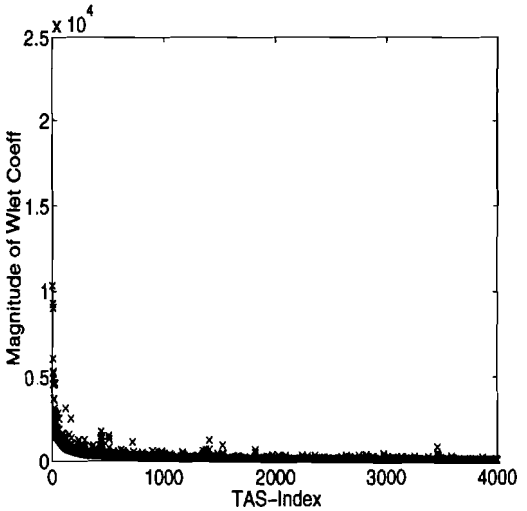
Fig. B.54. "bank" image with Daubechies D4 Wavelet.



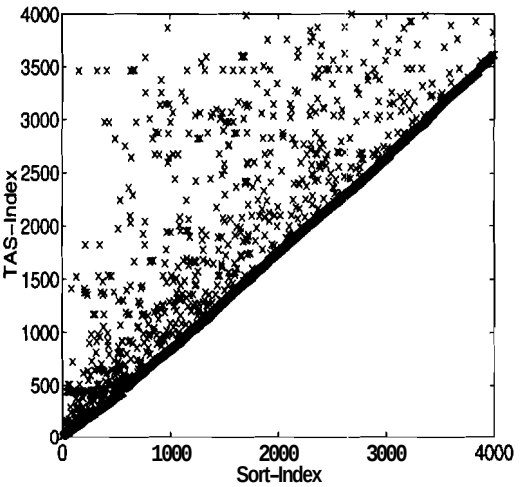
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

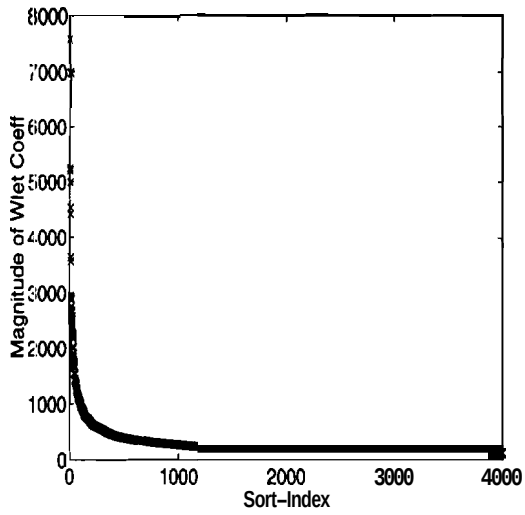


(c) Magnitudes vs TAS-Index

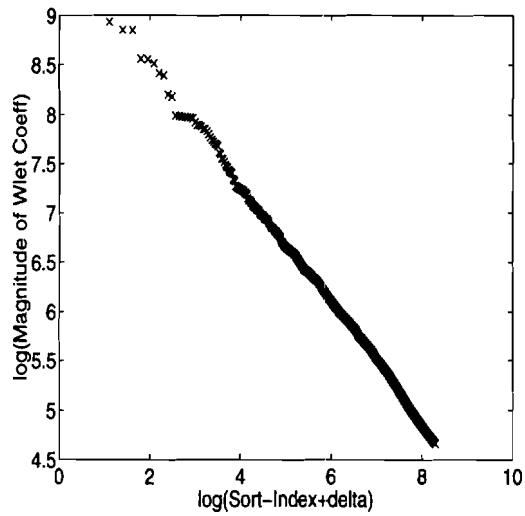


(d) TAS-Index vs Sort-Index

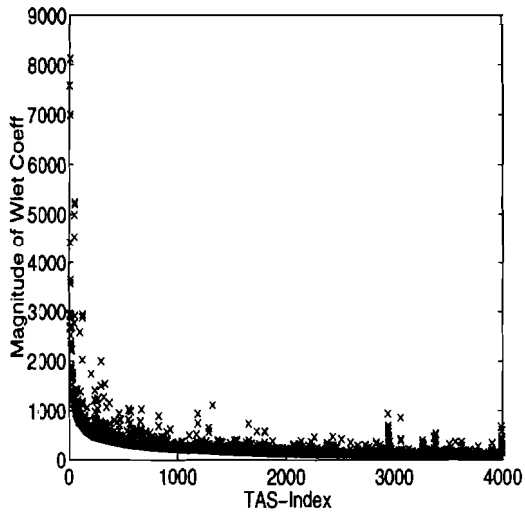
Fig. B.55. "bank" image with Spline-2 Wavelet.



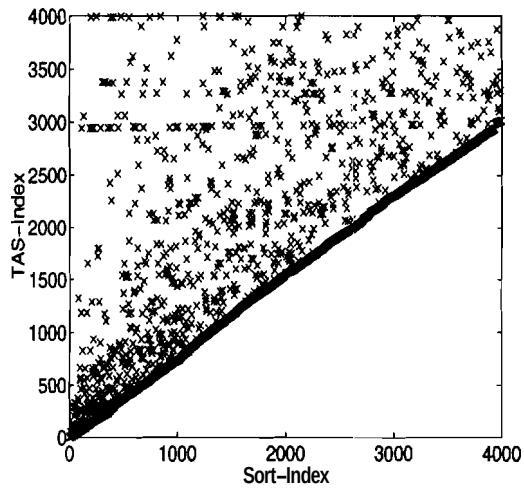
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

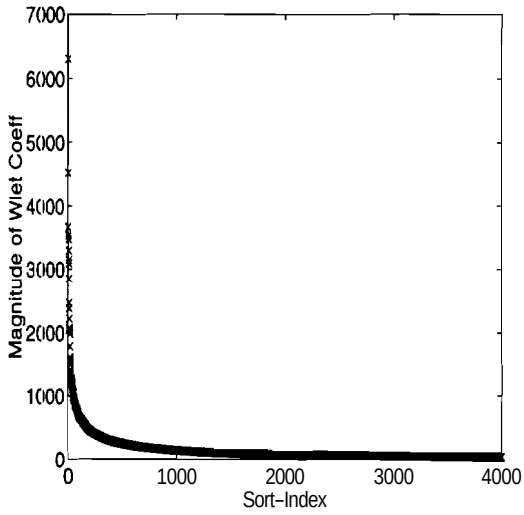


(c) Magnitudes vs TAS-Index

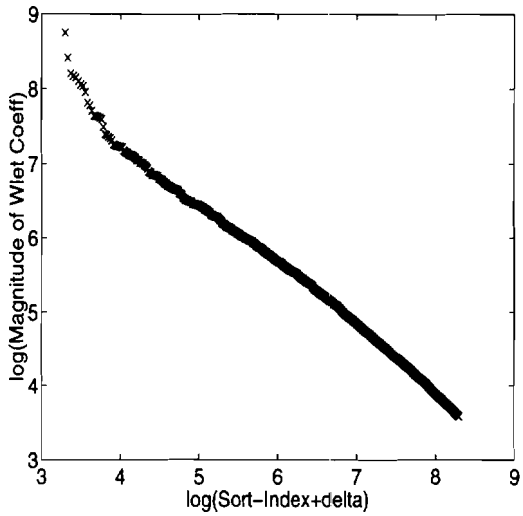


(d) TAS-Index vs Sort.-Index

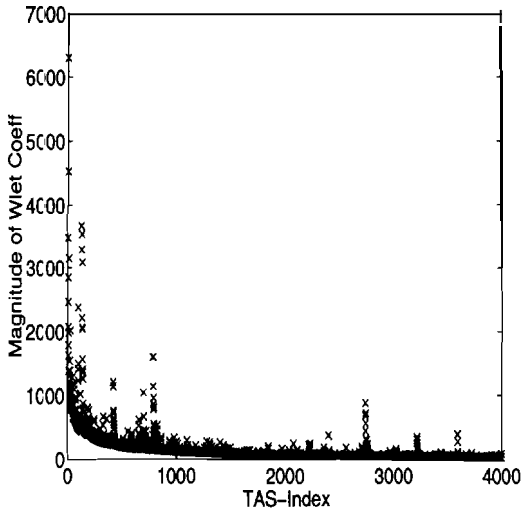
Fig. B.56. "bank" image with spline-variant Wavelet.



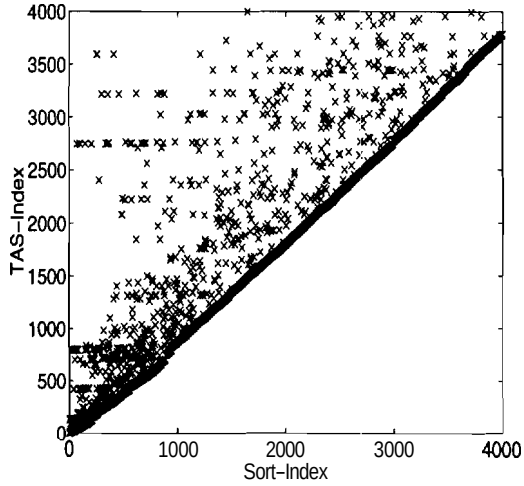
(a) Magnitudes vs Sort-Index



(b) log-log plot of **(a)**

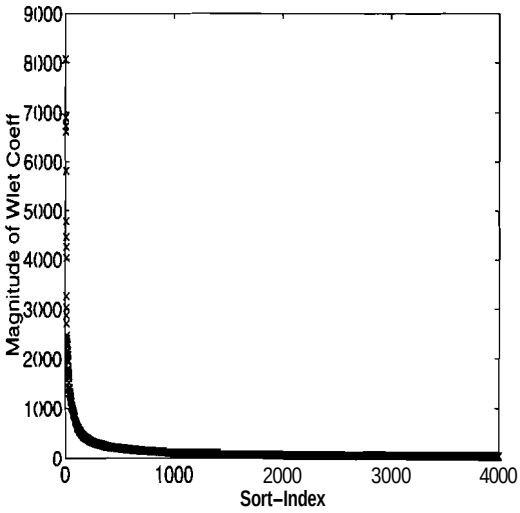


(c) Magnitudes vs TAS-Index

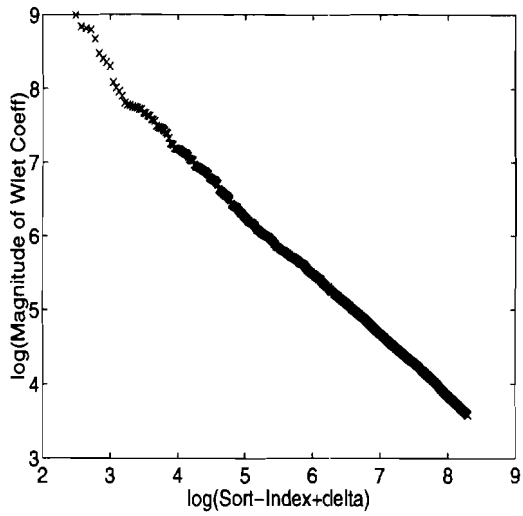


(d) TAS-Index vs Sort-Index

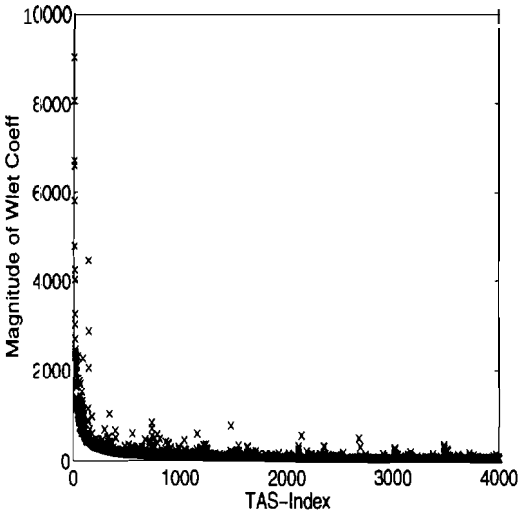
Fig. B.57. "U-Chart" image with Haar Wavelet.



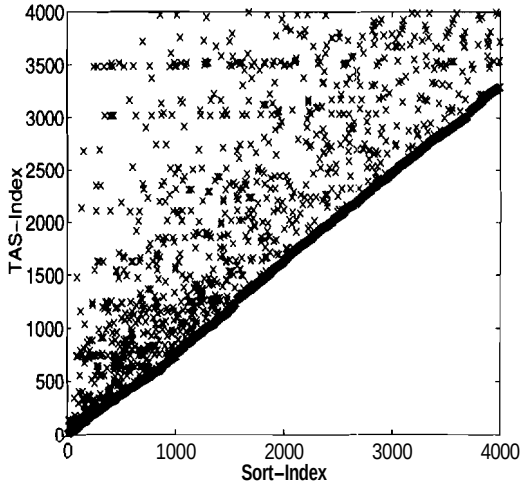
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

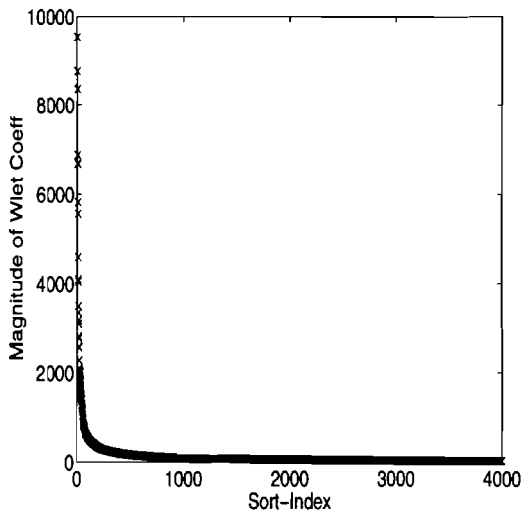


(c) Magnitudes vs TAS-Index

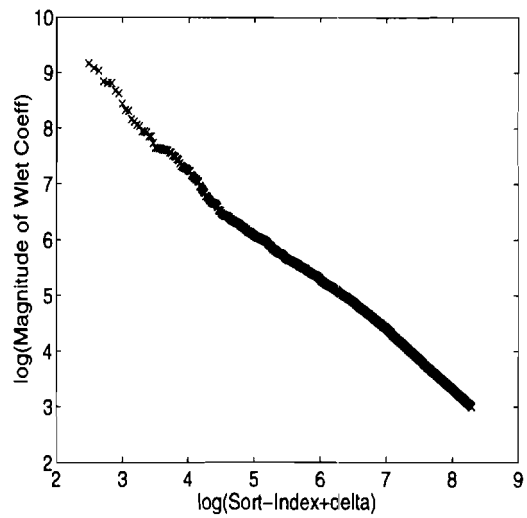


(d) TAS-Index vs Sort-Index

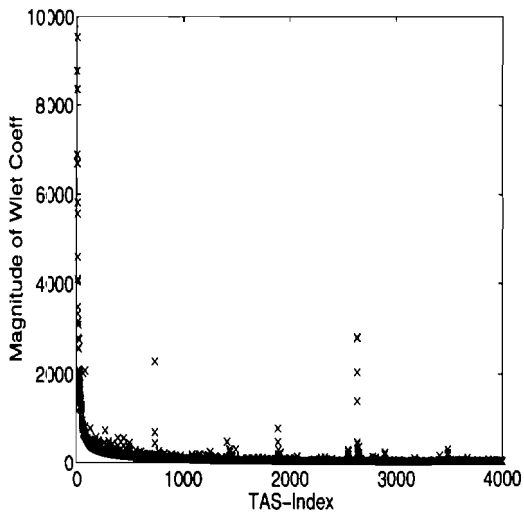
Fig. B.58. "U-Chart" image with Daubechies D4 Wavelet.



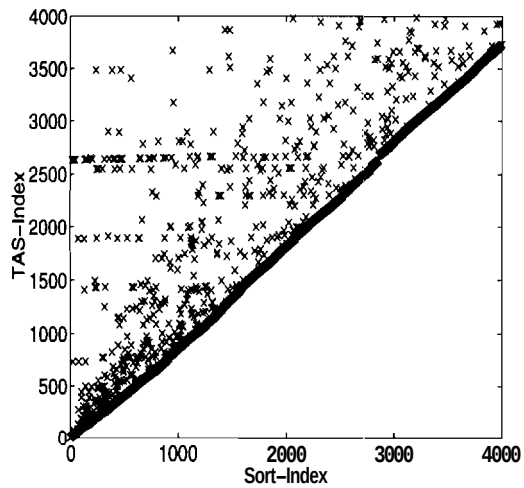
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

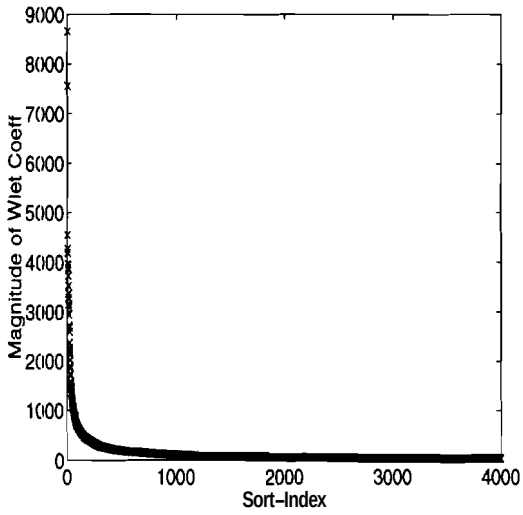


(c) Magnitudes vs TAS-Index

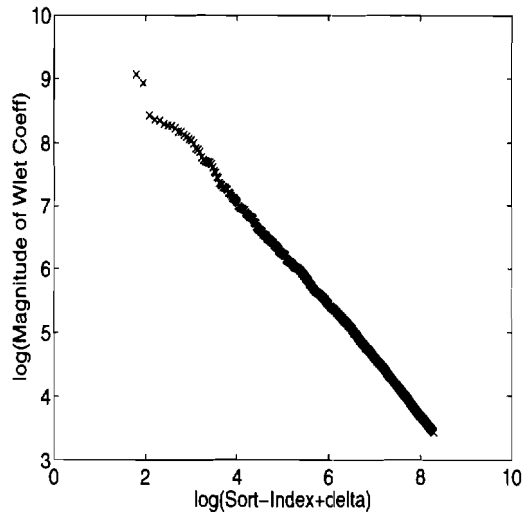


(d) TAS-Index vs Sort-Index

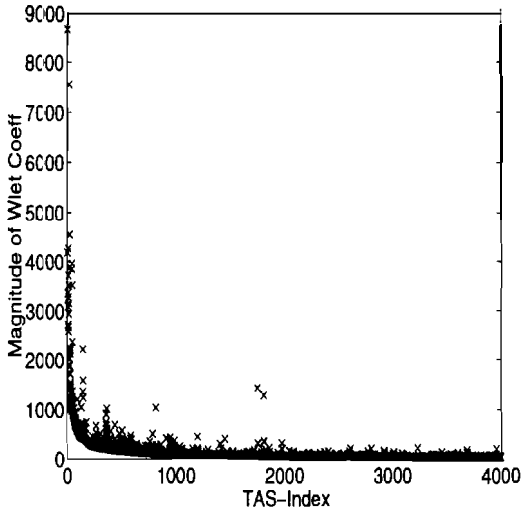
Fig. B.59. "U-Chart" image with Spline-2 Wavelet.



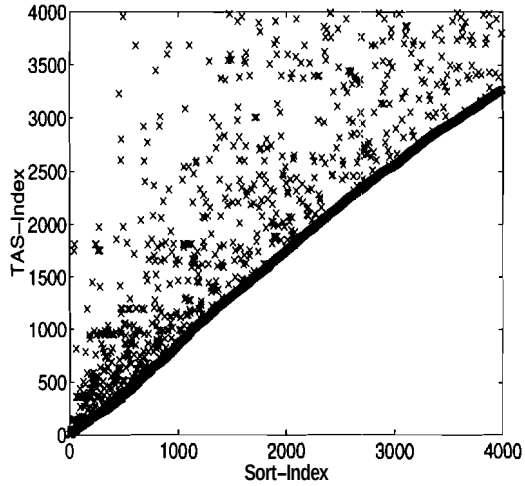
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

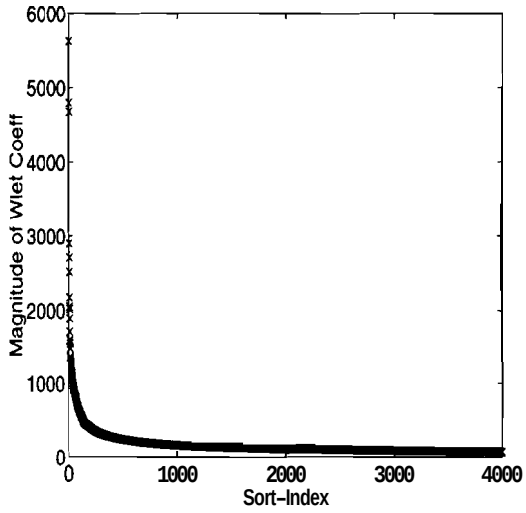


(c) Magnitudes vs TAS-Index

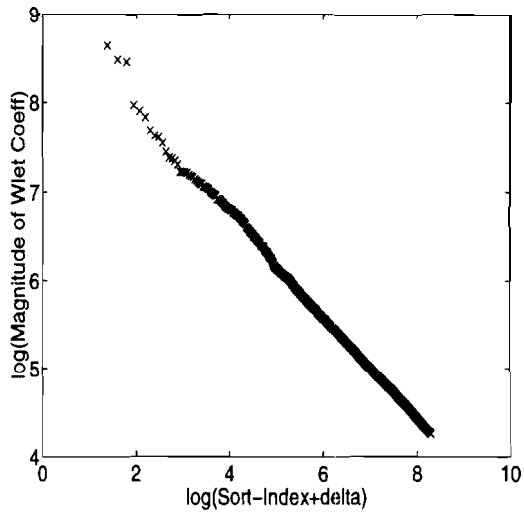


(d) TAS-Index vs Sort-Index

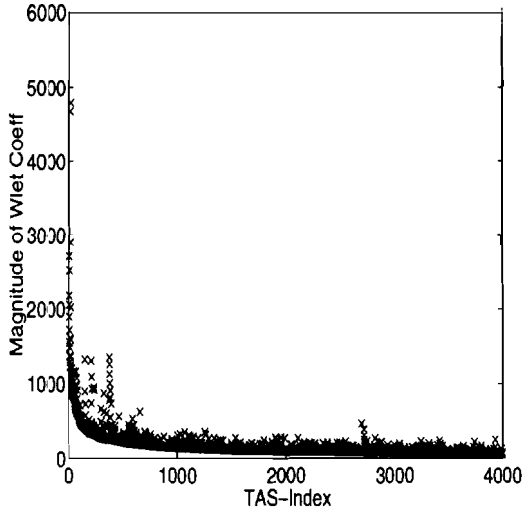
Fig. B.60. "U-Chart" image with spline-variant Wavelet.



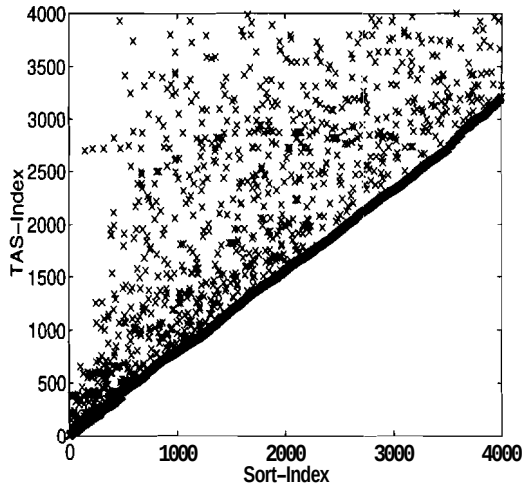
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

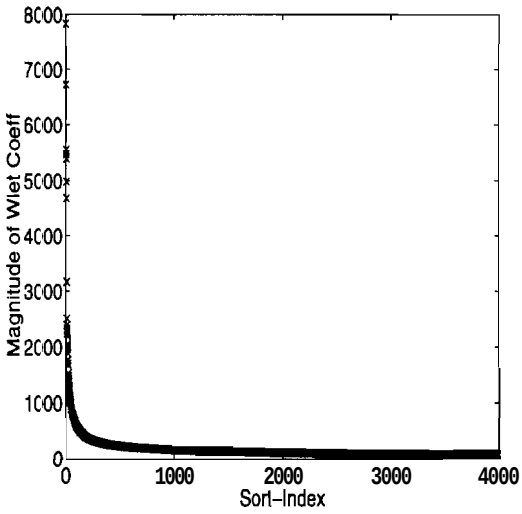


(c) Magnitudes vs TAS-Index

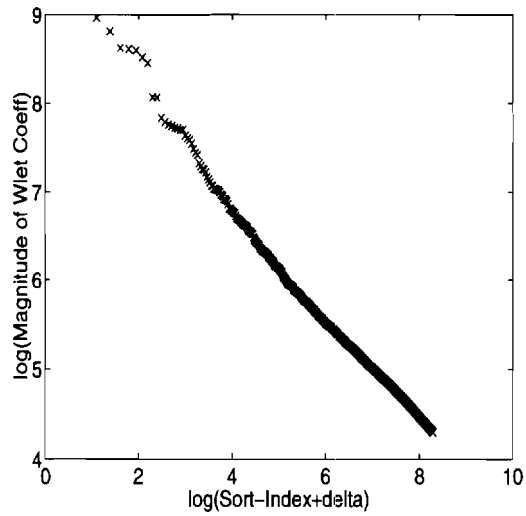


(d) TAS-Index vs Sort-Index

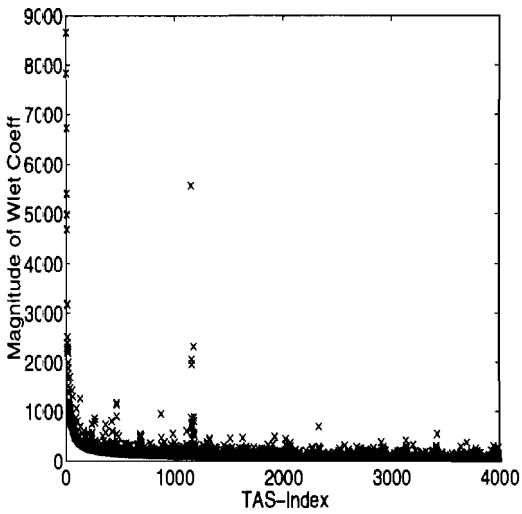
Fig. B.61. “Picnic” image with Haar Wavelet.



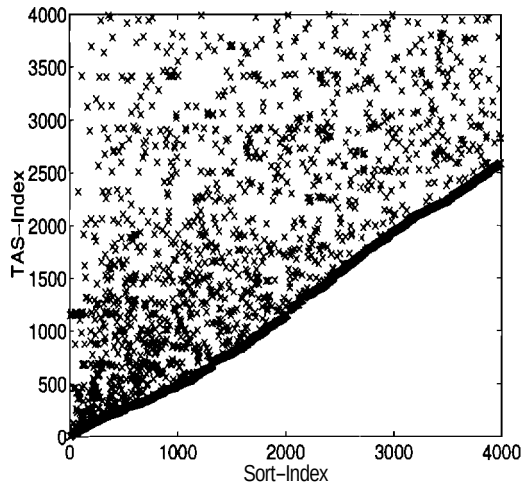
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

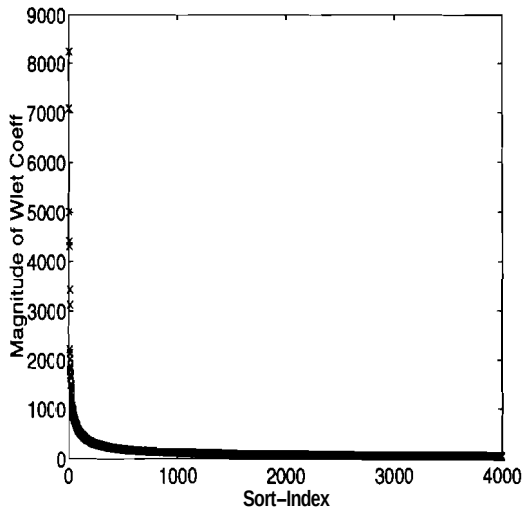


(c) Magnitudes vs TAS-Index

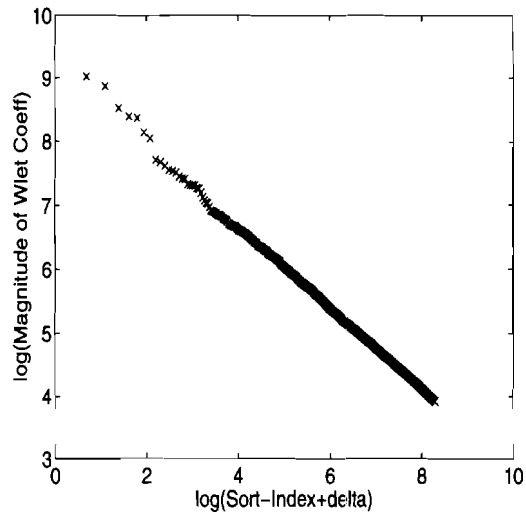


(d) TAS-Index vs Sort-Index

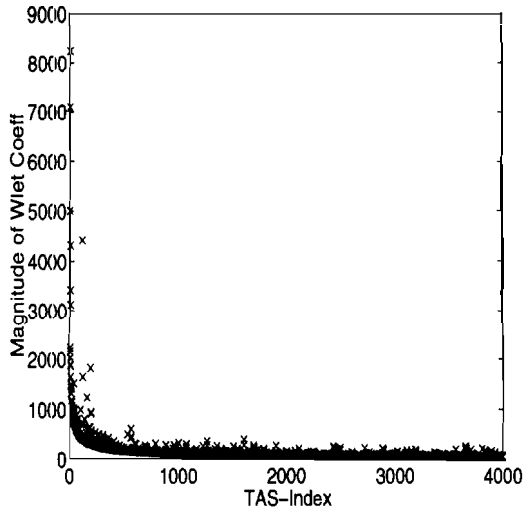
Fig. B.62. "Picnic" image with Daubechies D4 Wavelet,.



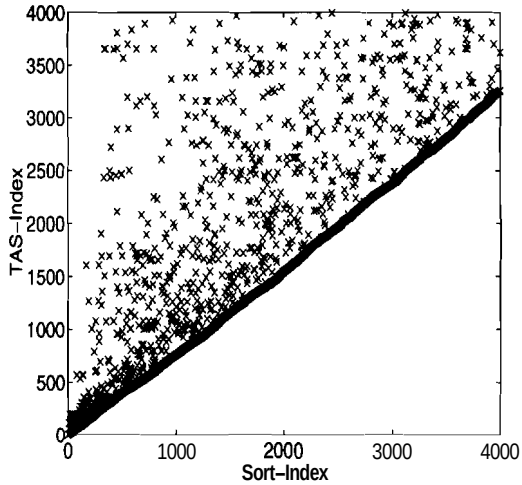
(a) Magnitudes vs Sort-Index



(b) log-log plot of **(a)**

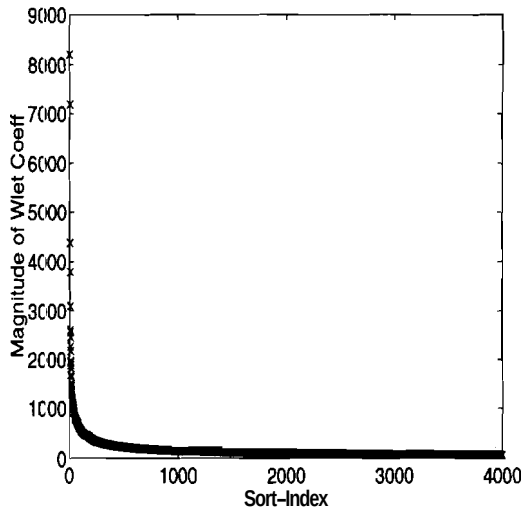


(c) Magnitudes vs TAS-Index

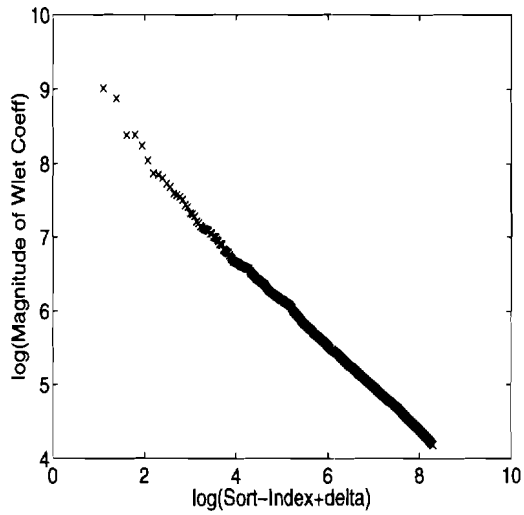


(d) TAS-Index vs Sort-Index

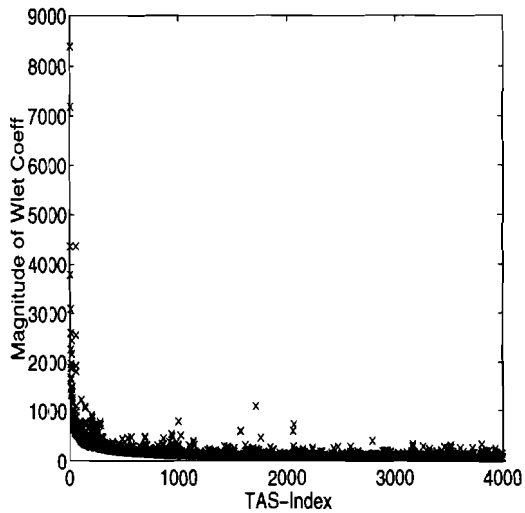
Fig. B.63. "Picnic" image with Spline-2 Wavelet.



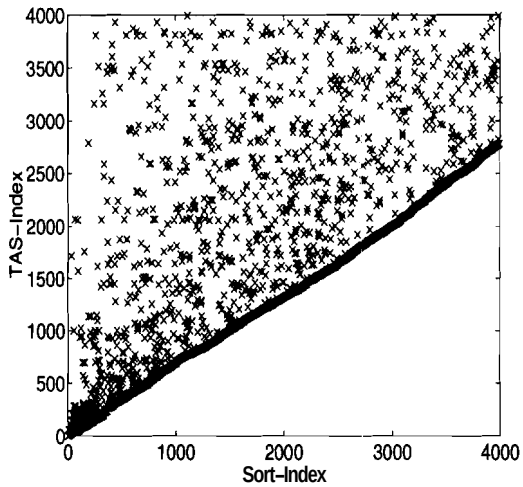
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

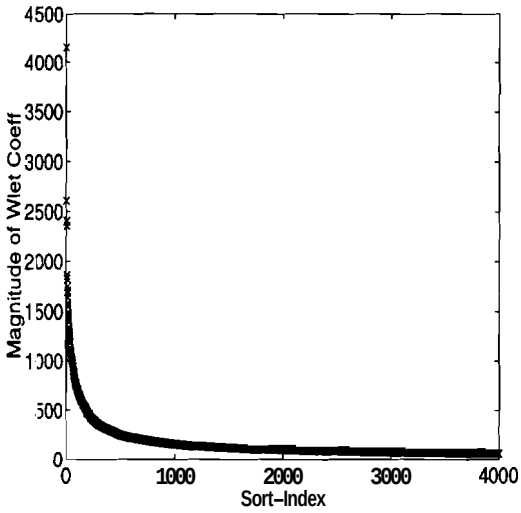


(c) Magnitudes vs TAS-Index

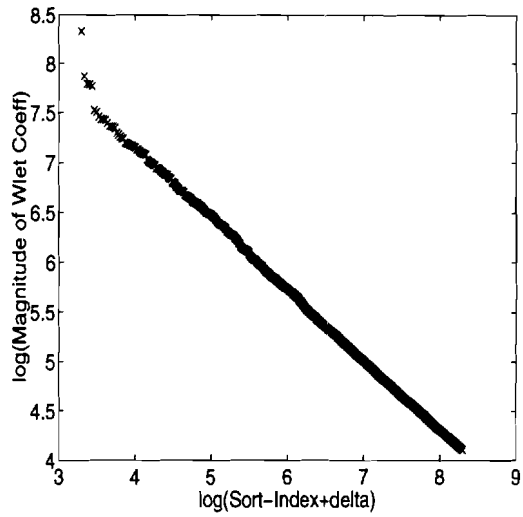


(d) TAS-Index vs Sort-Index

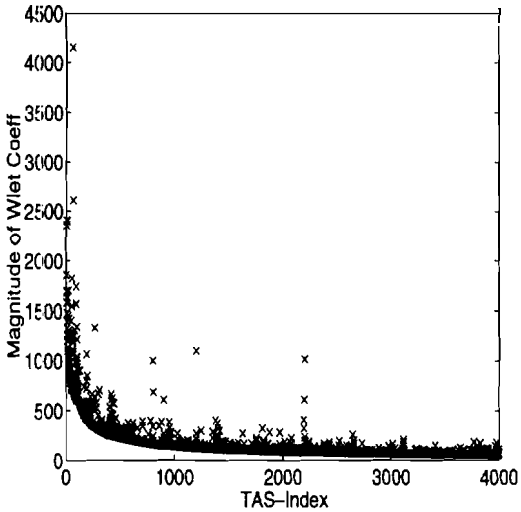
Fig. B.64. "Picnic" image with spline-variant Wavelet.



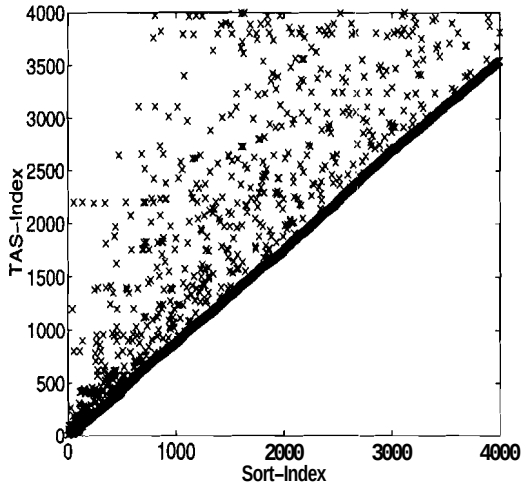
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

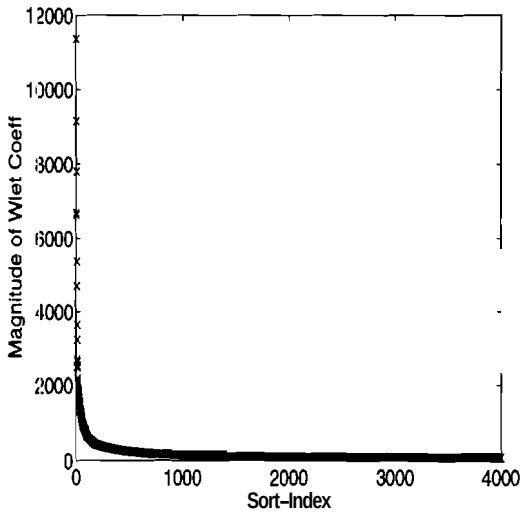


(c) Magnitudes vs TAS-Index

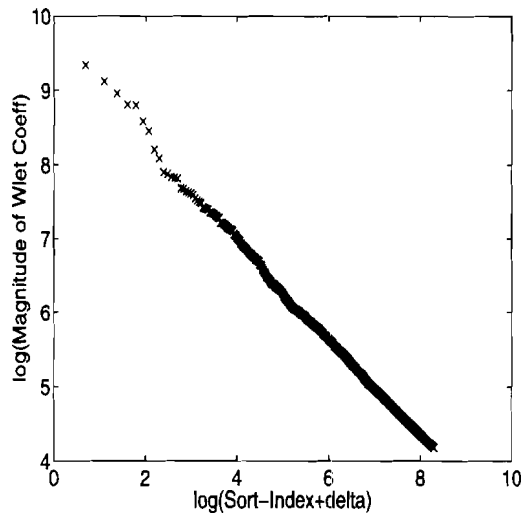


(d) TAS-Index vs Sort-Index

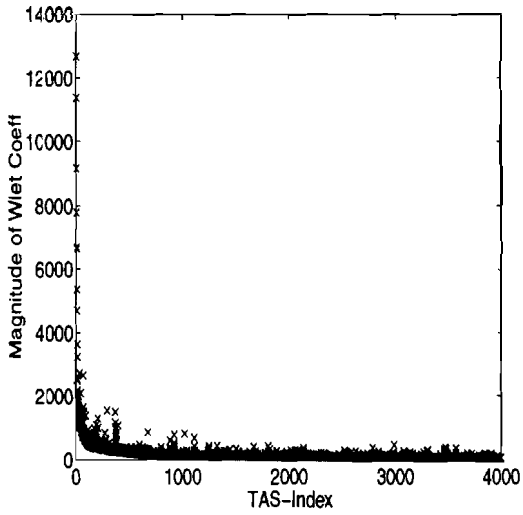
Fig. B.65. "Einstein" image with Haar Wavelet.



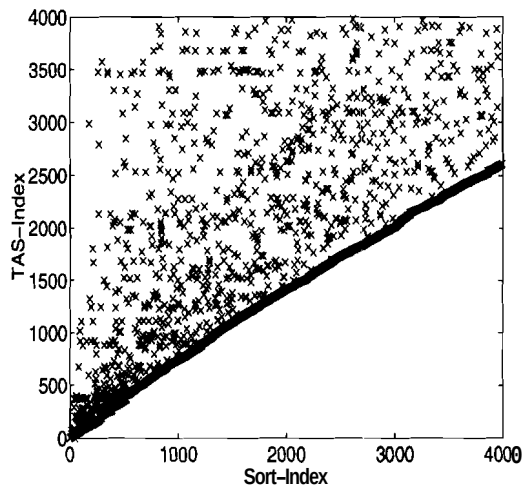
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

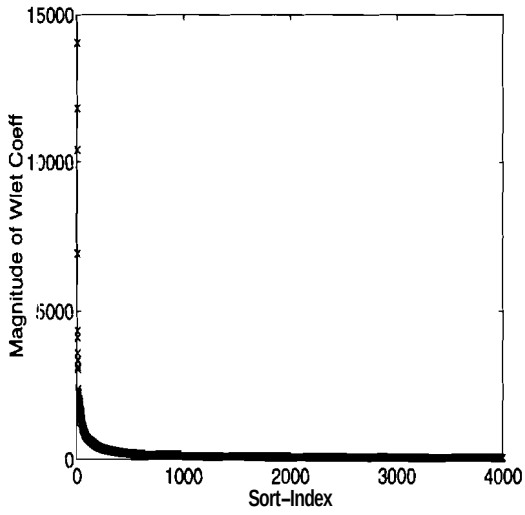


(c) Magnitudes vs TAS-Index

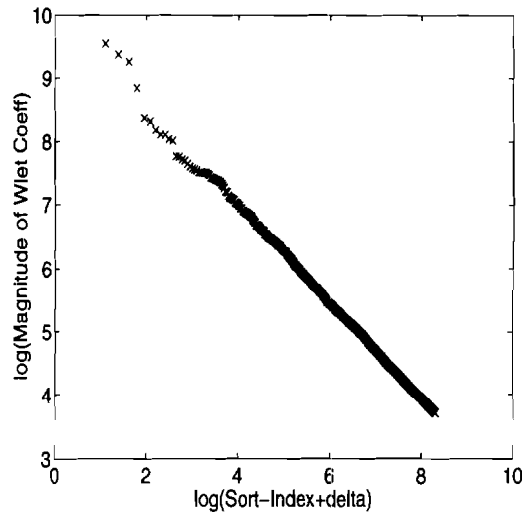


(d) TAS-Index vs Sort-Index

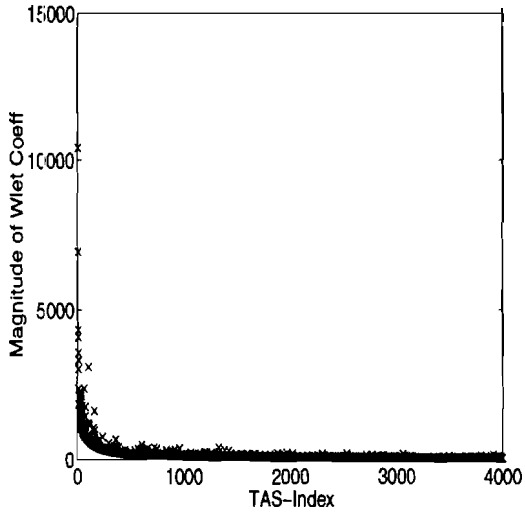
Fig. B.66. "Einstein" image with Daubechies D4 Wavelet.



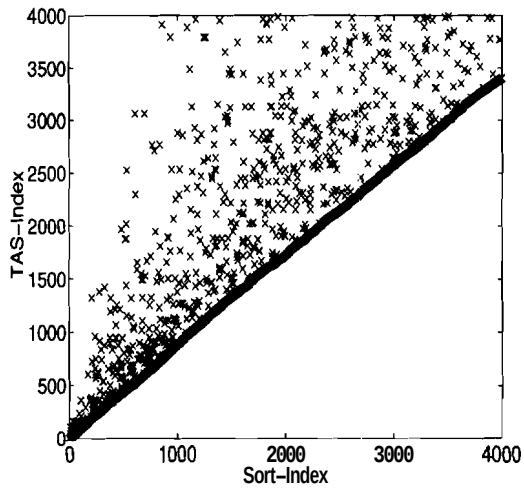
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

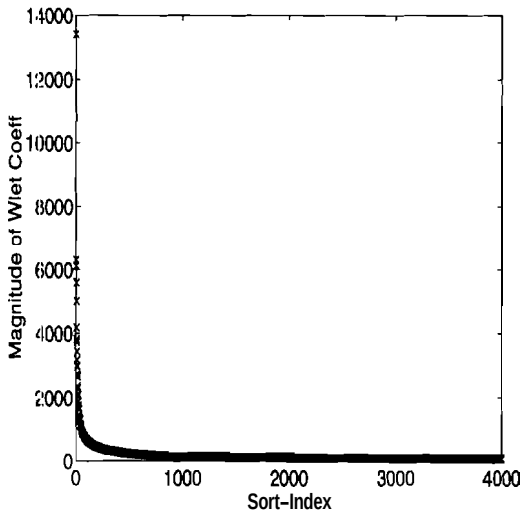


(c) Magnitudes vs TAS-Index

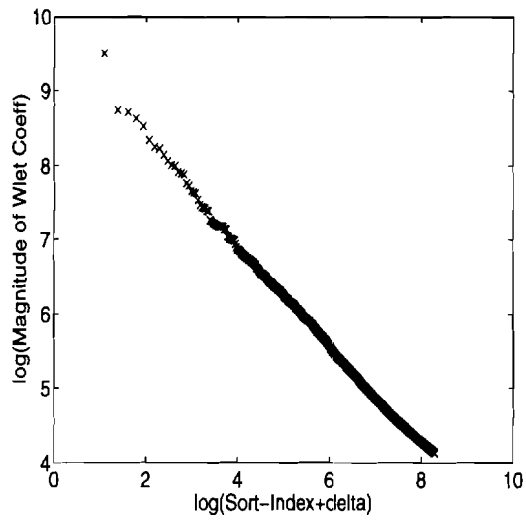


(d) TAS-Index vs Sort-Index

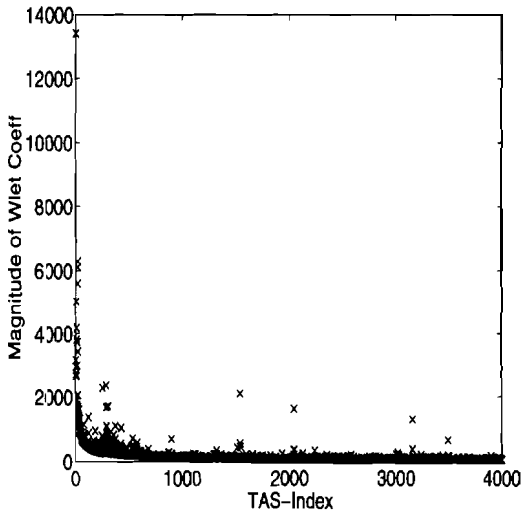
Fig. B.67. "Einstein" image with Spline-2 Wavelet.



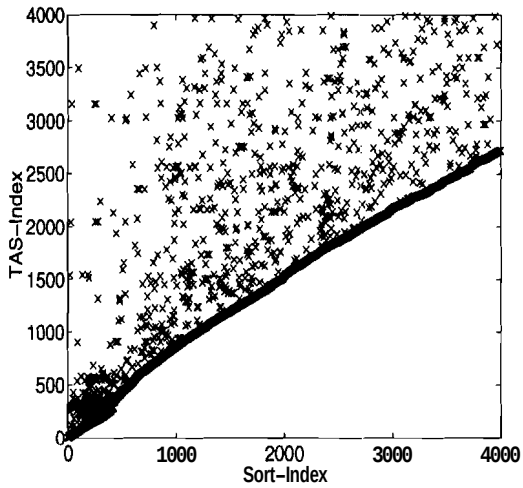
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

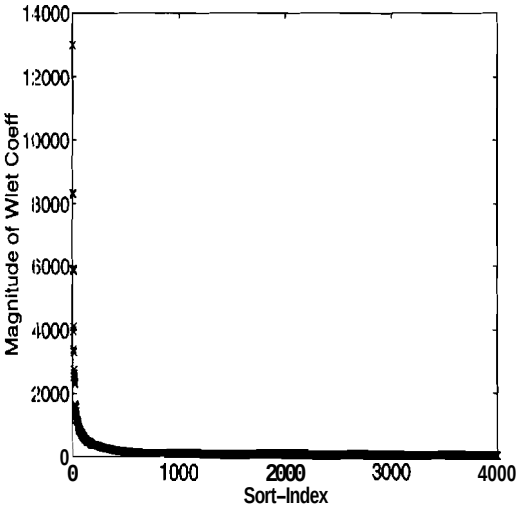


(c) Magnitudes vs TAS-Index

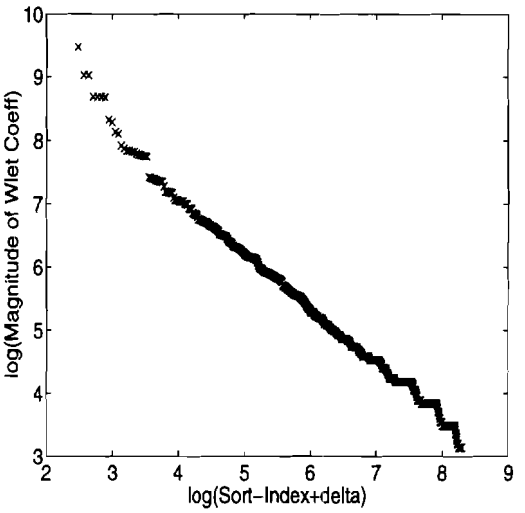


(d) TAS-Index vs Sort-Index

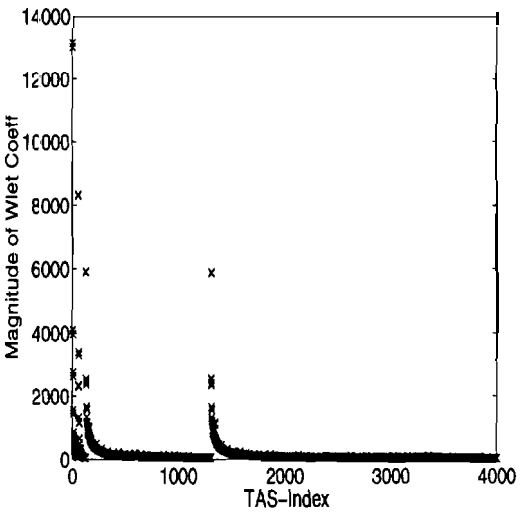
Fig. B.68. "Einstein" image with spline-variant Wavelet,.



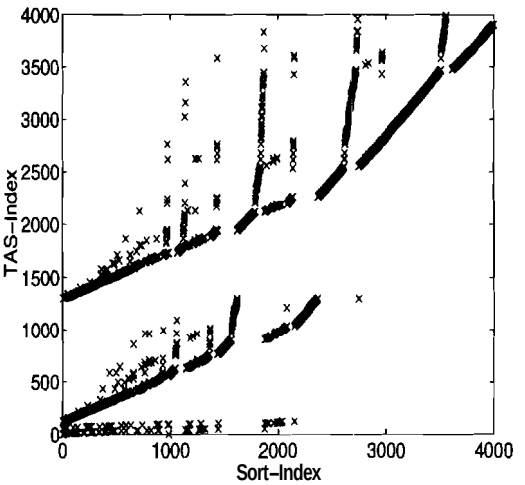
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

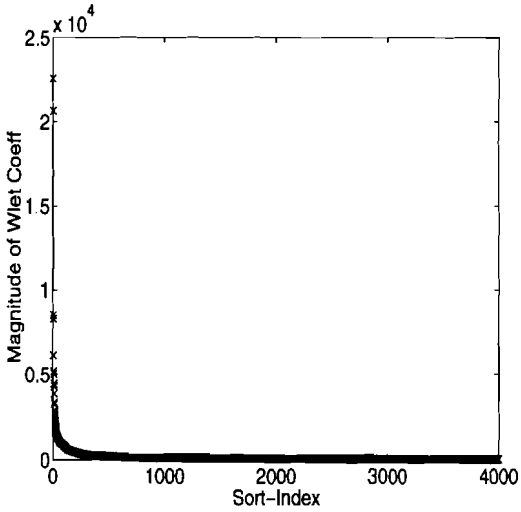


(c) Magnitudes vs TAS-Index

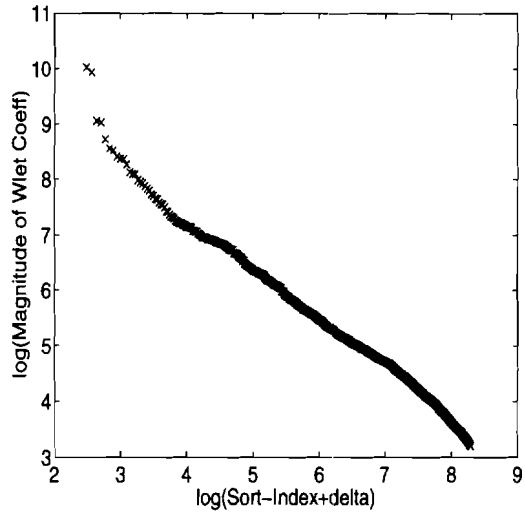


(d) TAS-Index vs Sort-Index

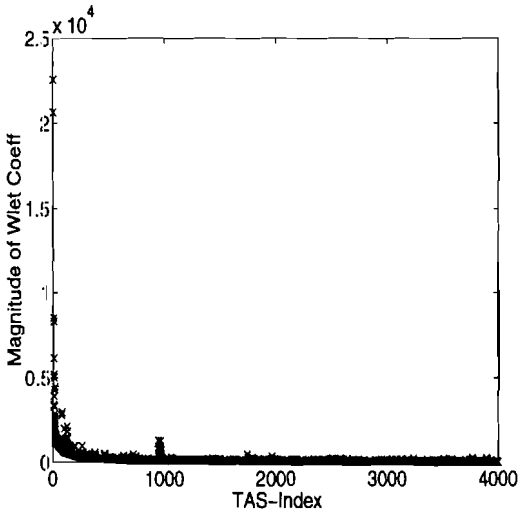
Fig. B.69. Synthetic image: “Oval” image with Haar Wavelet.



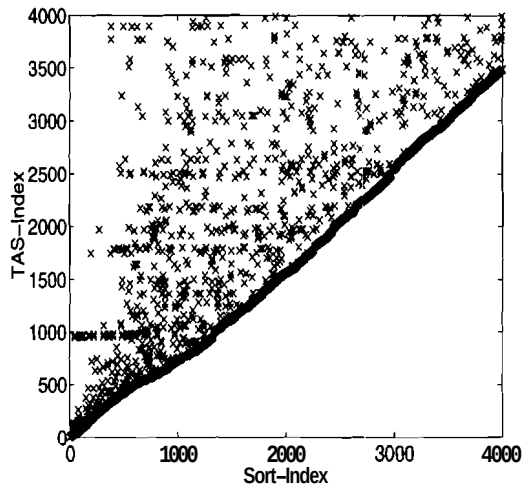
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

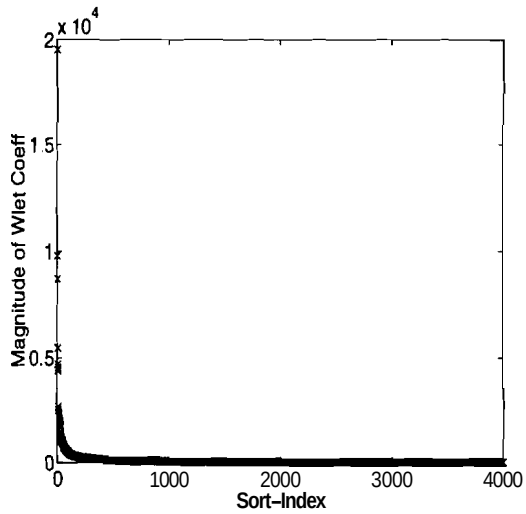


(c) Magnitudes vs TAS-Index

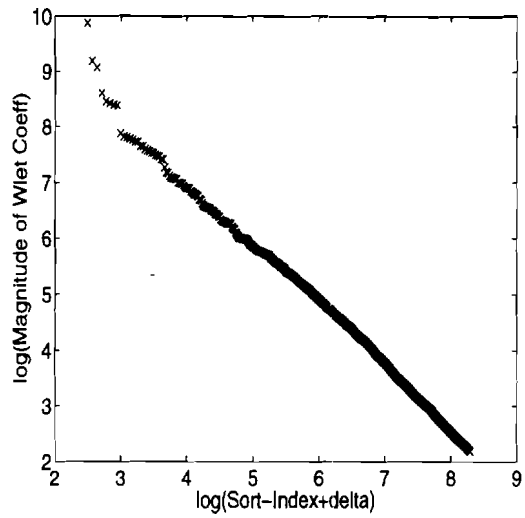


(d) TAS-Index vs Sort-Index

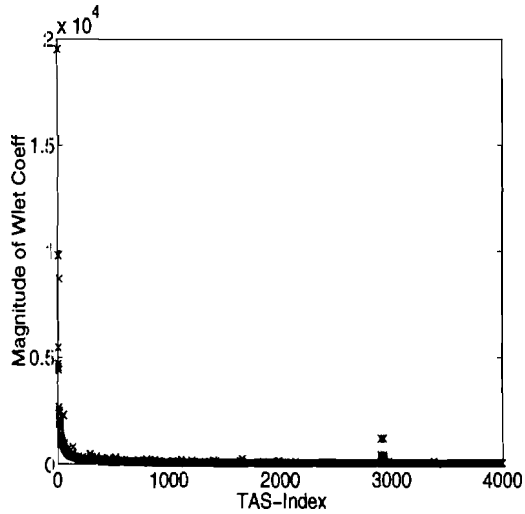
Fig. B.70. Synthetic image: "Oval" image with Daubechies D4 Wavelet.



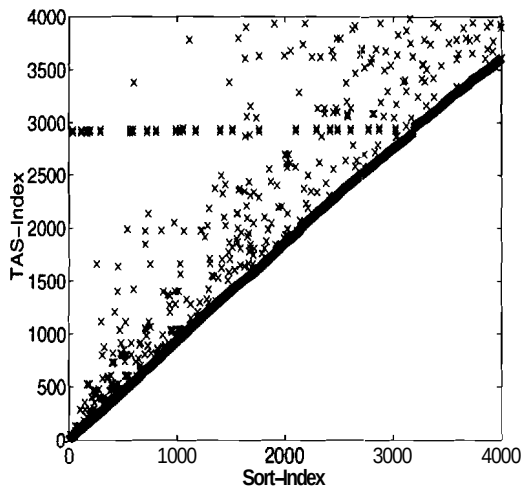
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

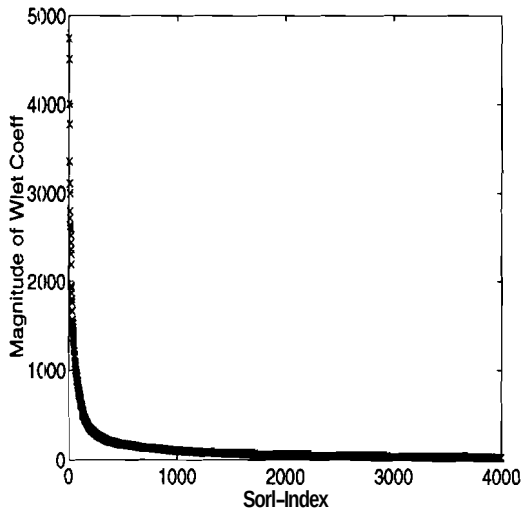


(c) Magnitudes vs TAS-Index

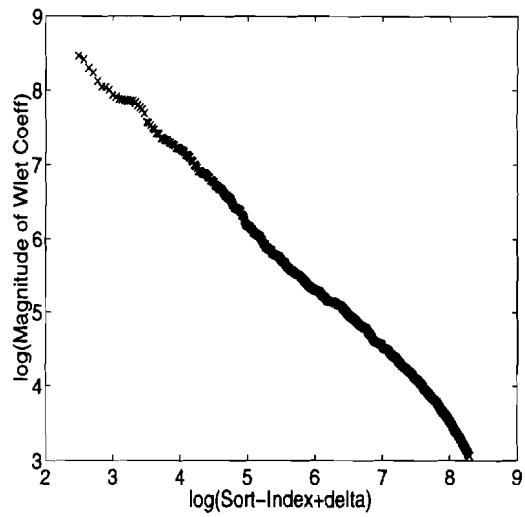


(d) TAS-Index vs Sort-Index

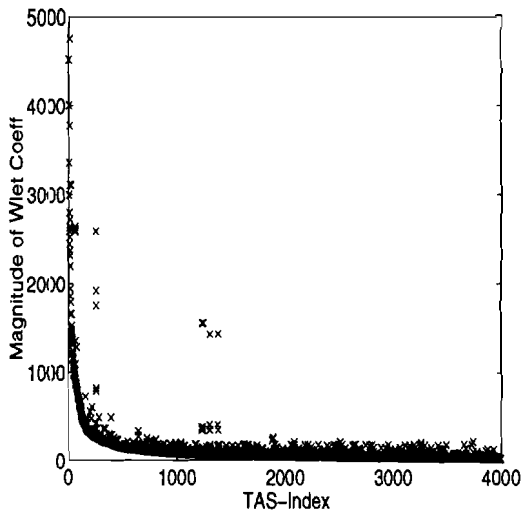
Fig. B.71. Synthetic image: "Oval" image with Spline-2 Wavelet.



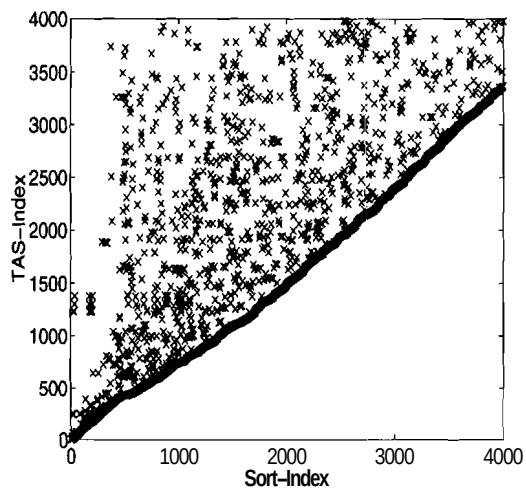
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

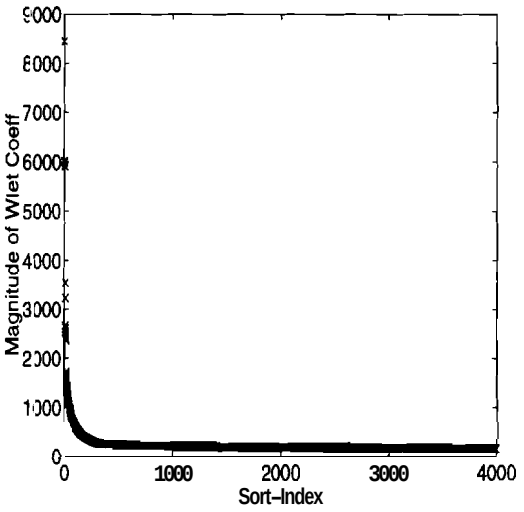


(c) Magnitudes vs TAS-Index

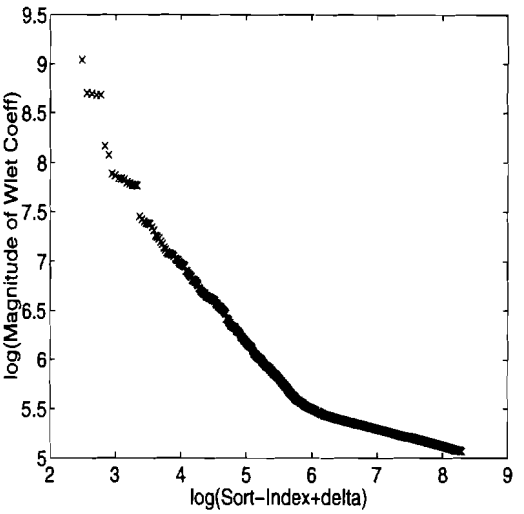


(d) TAS-Index vs Sort-Index

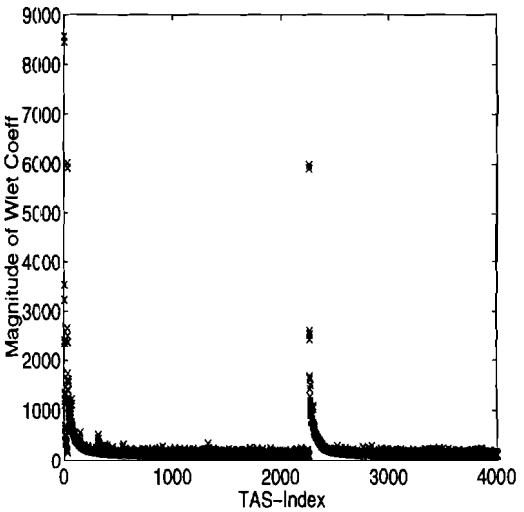
Fig. B.72. Synthetic image: "Oval" image with spline-variant Wavelet.



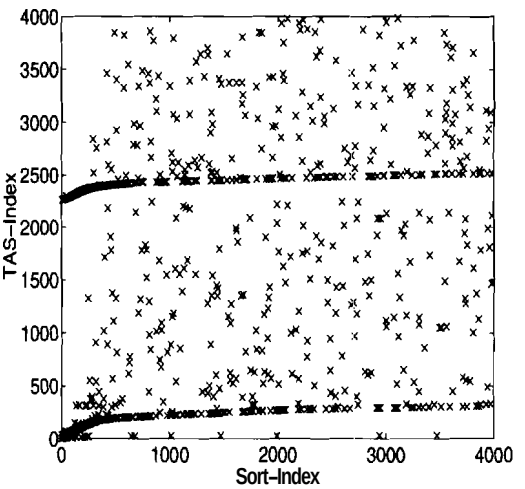
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

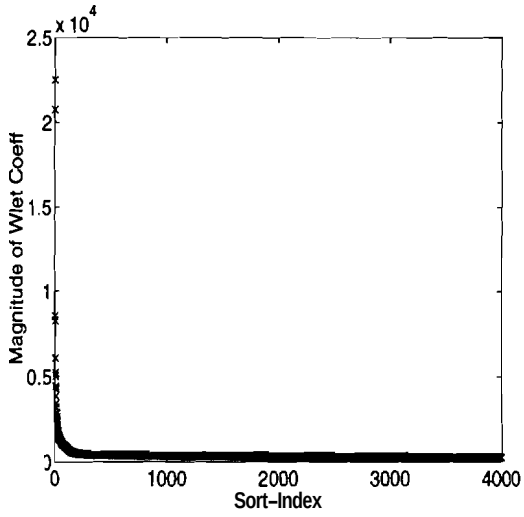


(c) Magnitudes vs TAS-Index

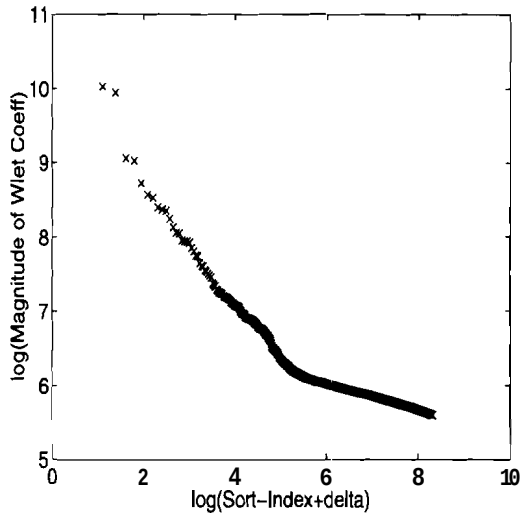


(d) TAS-Index vs Sort-Index

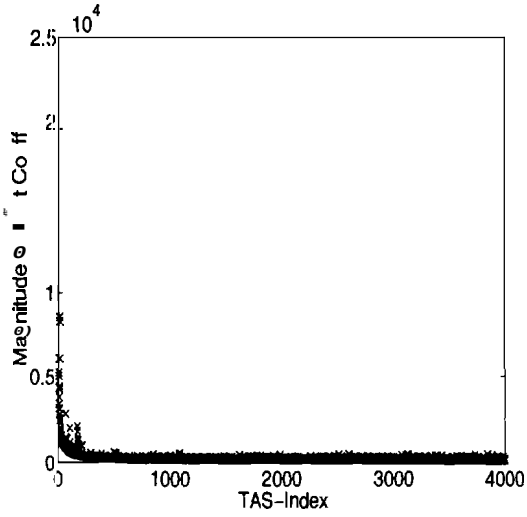
Fig. B.73. Synthetic image: "Oval+Noise" image with Haar Wavelet.



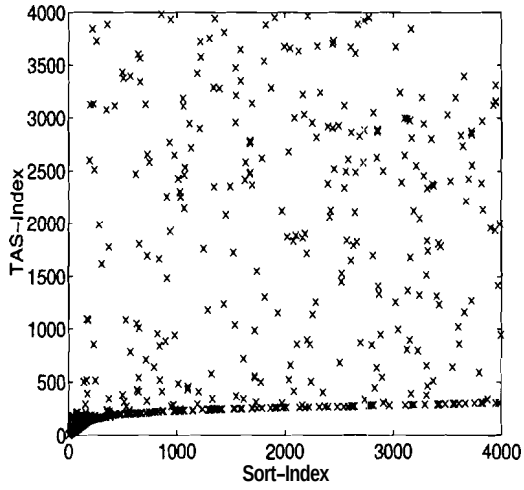
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

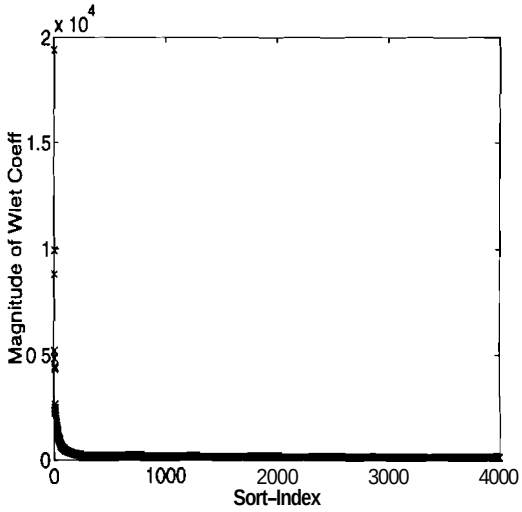


(c) Magnitudes vs TAS-Index

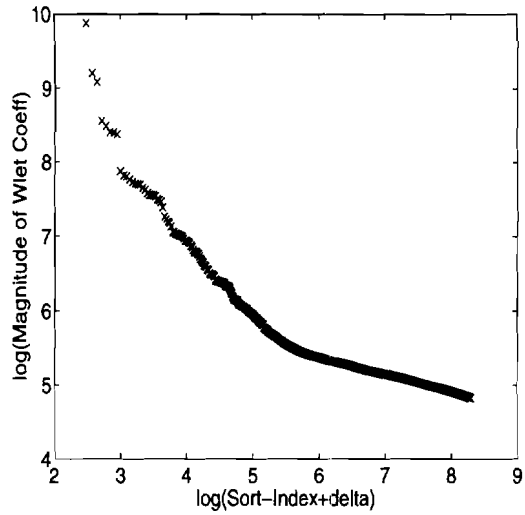


(d) TAS-Index vs Sort-Index

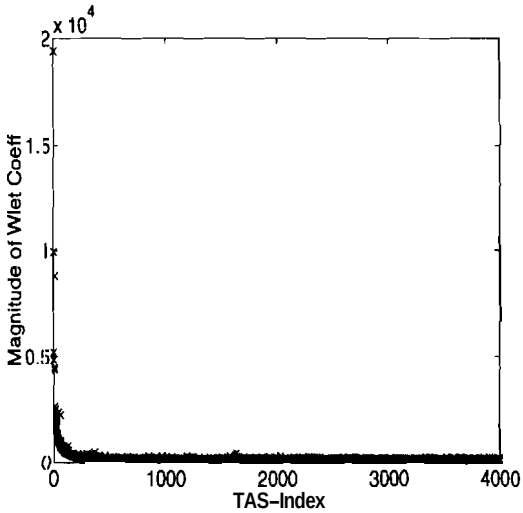
Fig. B.74. Synthetic image: "Oval+Noise" image with Daubechies D4 Wavelet.



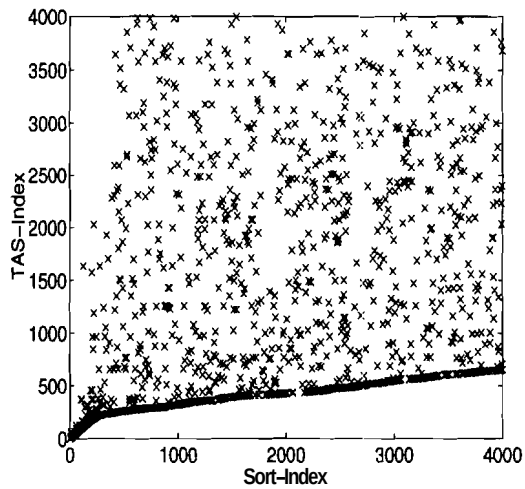
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

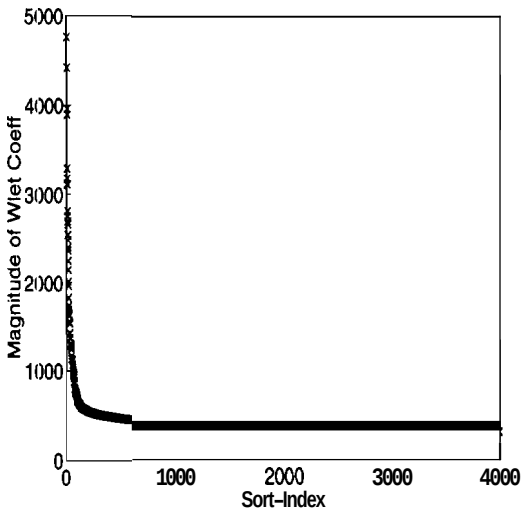


(c) Magnitudes vs TAS-Index

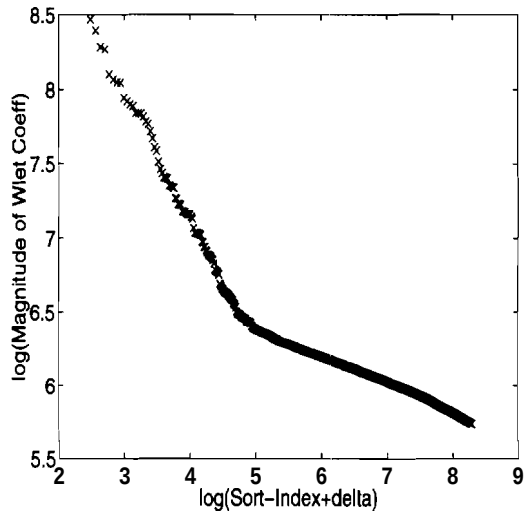


(d) TAS-Index vs Sort-Index

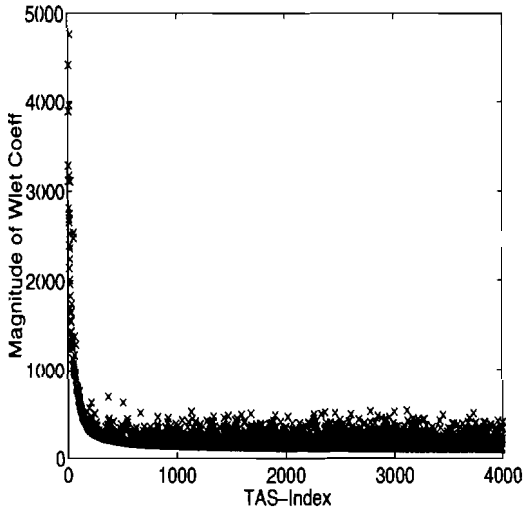
Fig. B.75. Synthetic image: “Oval+Noise” image with Spline-2 Wavelet.



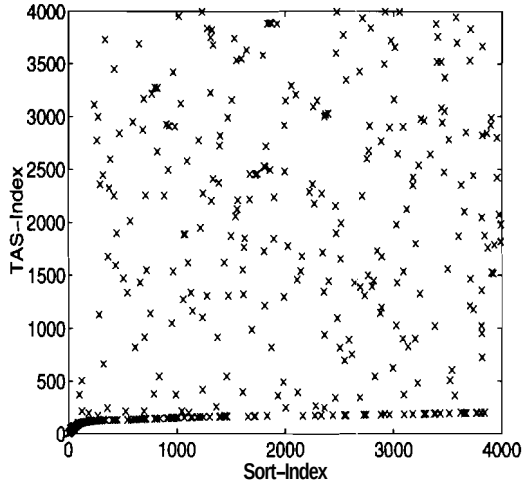
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

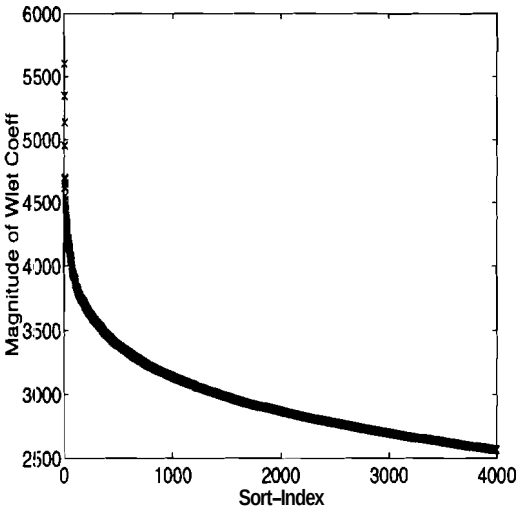


(c) Magnitudes vs TAS-Index

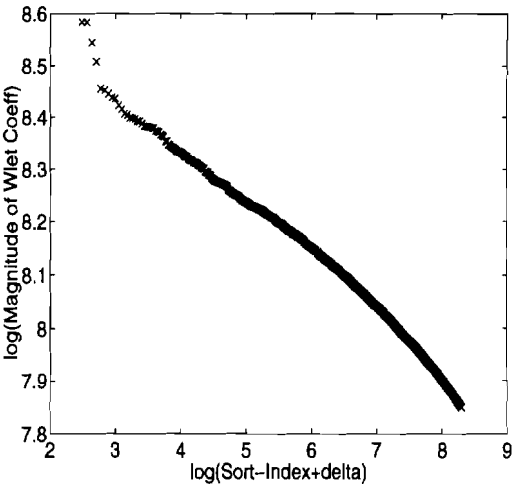


(d) TAS-Index vs Sort-Index

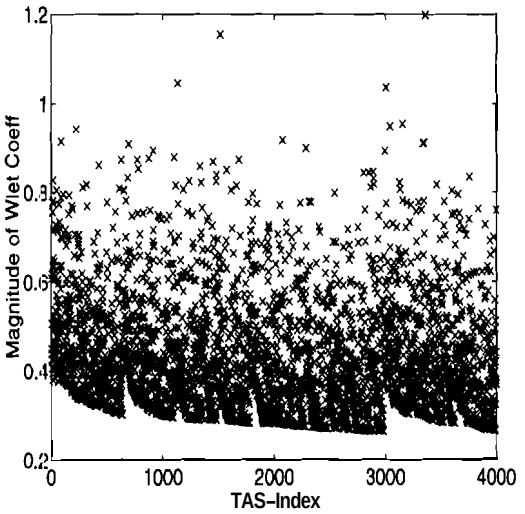
Fig. B.76. Synthetic image: "Oval+Noise" image with spline-variant Wavelet.



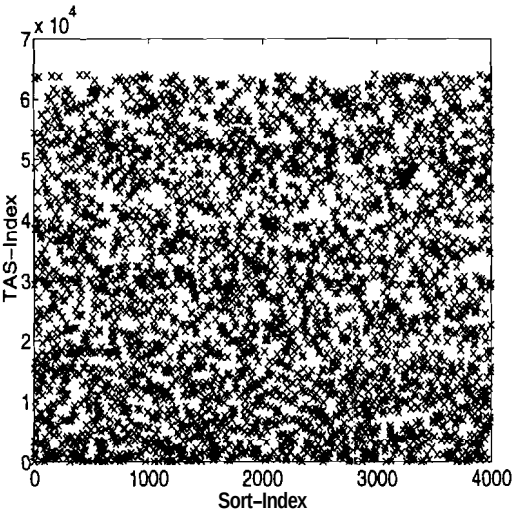
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

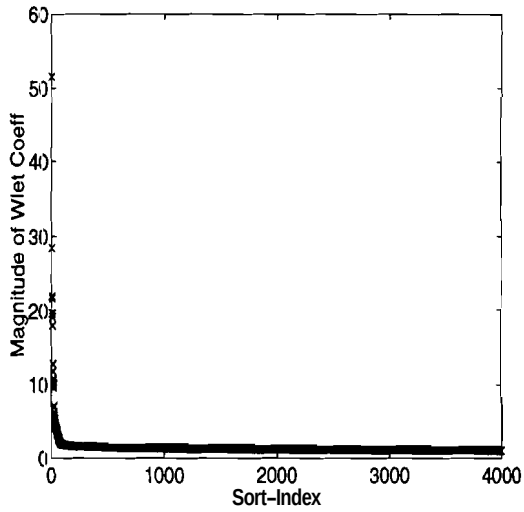


(c) Magnitudes vs TAS-Index

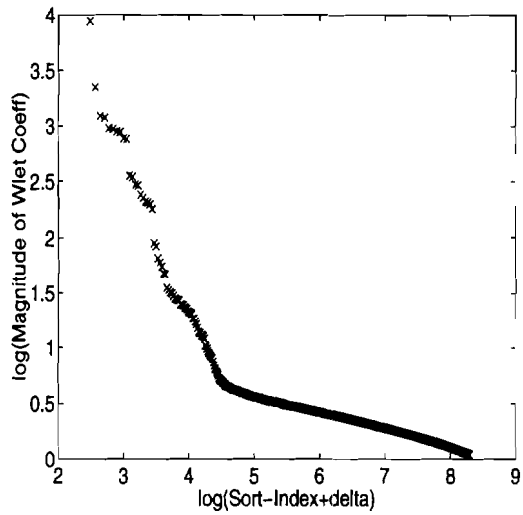


(d) TAS-Index vs Sort-Index

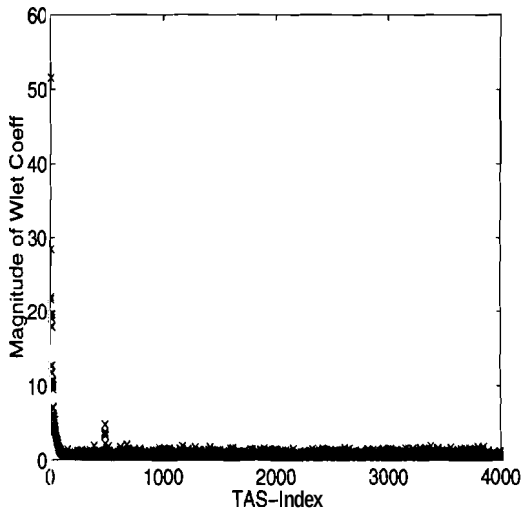
Fig. B.77. Synthetic image: "Noise" image with Haar Wavelet.



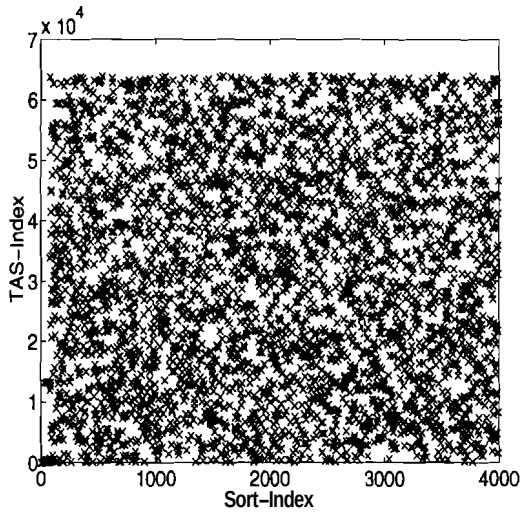
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

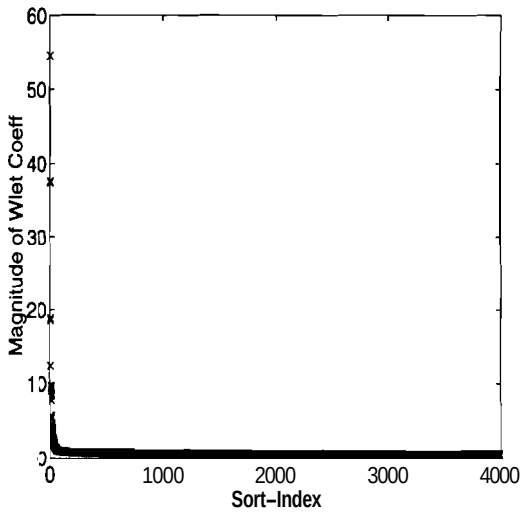


(c) Magnitudes vs TAS-Index

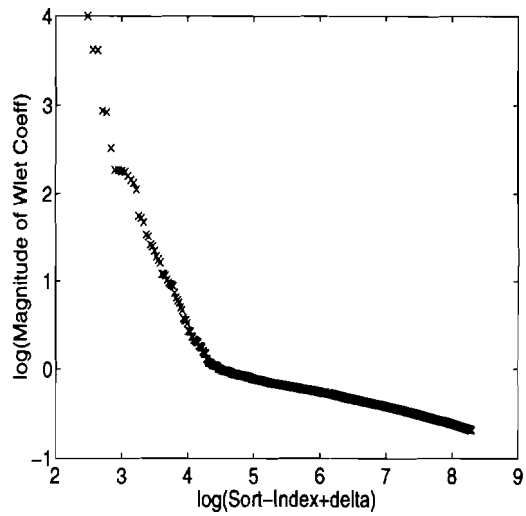


(d) TAS-Index vs Sort-Index

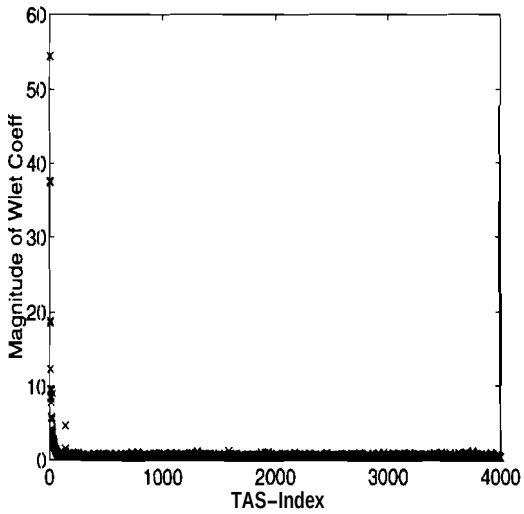
Fig. B.78. Synthetic image: "Noise" image with Daubechies D4 Wavelet.



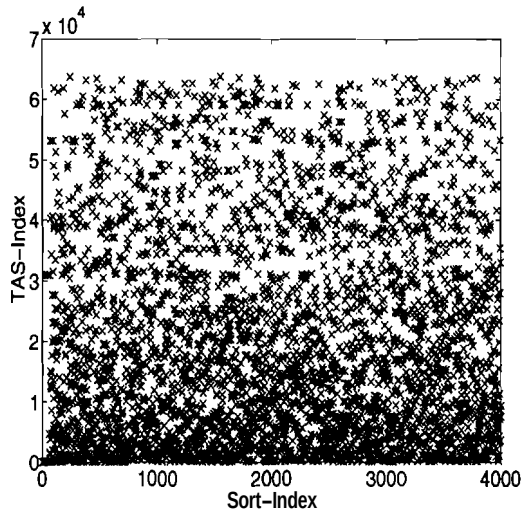
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)

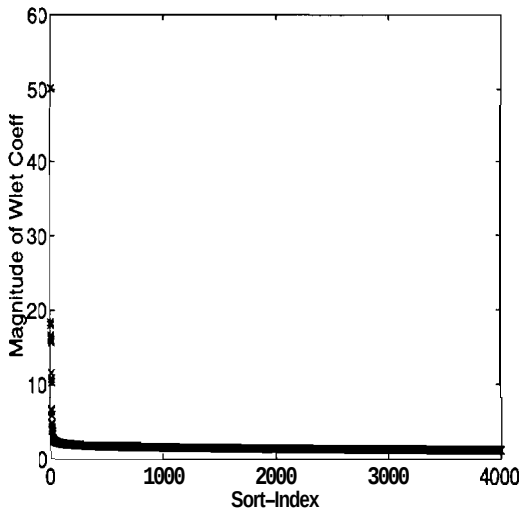


(c) Magnitudes vs TAS-Index

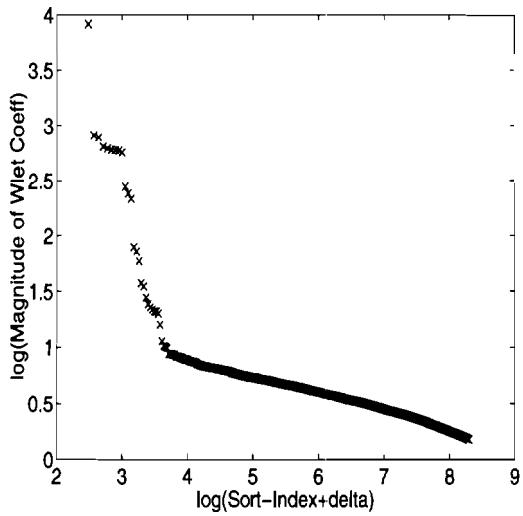


(d) TAS-Index vs Sort-Index

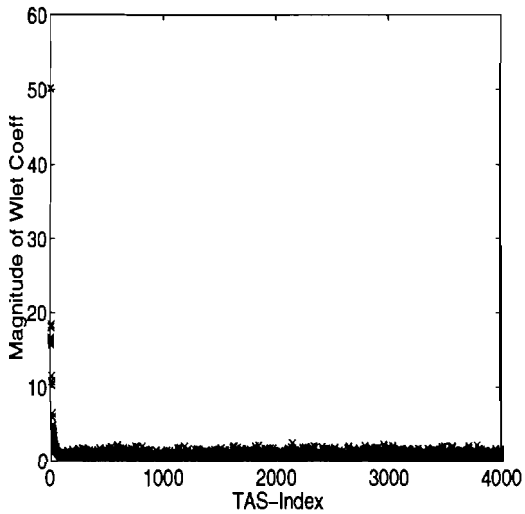
Fig. B.79. Synthetic image: "Noise" image with Spline-:! Wavelet.



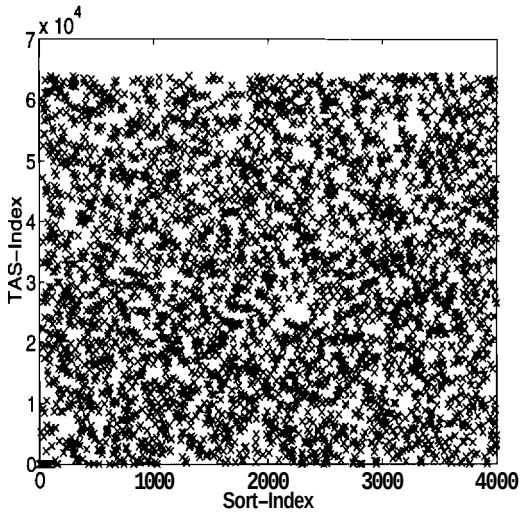
(a) Magnitudes vs Sort-Index



(b) log-log plot of (a)



(c) Magnitudes vs TAS-Index



(d) TAS-Index vs Sort-Index

Fig. B.80. Synthetic image: "Noise" image with spline-variant Wavelet.